Tivoli. software

IBM

Redpaper

Ling Tai
Ron Baker
Elizabeth Edmiston
Ben Jeffcoat

IBM Tivoli Common Data Model: Guide to Best Practices

The Common Data Model (CDM) is an information model that provides consistent definitions for managed resources, business systems and processes, and other data, and the relationships between those elements. CDM is based on the unified modeling language (UML). This IBM® Redpaper presents a set of example templates and scenarios that help users learn and apply the basics of the Common Data Model. Each scenario identifies the CDM classes that are used and defines the important relationships among these classes. This Redpaper can serve as a reference for IT specialists implementing Common Data Model as part of a Tivoli® Systems Management Solution.

Introduction

Many customers use multiple IBM Tivoli management products in a single IT enterprise. Each product maintains its own separate data that is related to the set of resources it manages. As a result, the data is frequently duplicated, both logically and physically. Data center servers, in particular, tend to be defined multiple times, along with their related infrastructure. Application information is another type of data that is often duplicated in many products.

The Tivoli data integration initiative seeks to address this problem by enabling the exchange of data among Tivoli management products. The goals of data integration are as follows:

- Cut the costs of data entry and discovery, thereby reducing mean time to value.
- Create data consistency among products, which simplifies the management environment and reduces errors caused by conflicting information.
- Enable administrators to find and link together information across products, providing a more accurate and complete picture for analysis and reporting.

Data integration is possible only when there is a clear understanding of what is being integrated — from the basic semantics that define a resource, to resource attributes and the relationships among resources. The Common Data Model is an information model that provides this understanding. When managed resources and business components are modeled using CDM specifications, Tivoli management products can understand and more easily exchange data across the enterprise.

This IBM Redpaper presents a set of example templates and scenarios that help users learn and apply the basics of the CDM. Each scenario identifies the CDM classes that are used and defines the important relationships among these classes. Initially, users can focus on the classes and relationships covered in scenarios that are applicable to them. As they gain proficiency with the CDM, they can extend or simplify the example templates to model their own specific environments.

Key terms and concepts

Table 1 lists and defines terms and concepts that are important to an understanding of the CDM.

Table 1 Key terms and concepts

Term/concept	**Definition**
IT Infrastructure Library® (ITIL®)	A library of best-practice documents explaining how companies run their IT operations. ITIL includes organization descriptions, process flows, and data descriptions for a standard set of management disciplines.
Managed element (or model object)	A managed element is the top of the CDM class hierarchy, representing all types of elements known to Tivoli management products, including the Configuration Management Database (CMDB). Managed elements include all configuration items (CIs), transaction profiles, business systems and processes, and other elements that are managed by the products.
Configuration item (CI)	A term from ITIL that represents a hardware or software asset under the control of the CMDB. A CI must be identifiable and tracked within the organization. It requires the Configuration Management process to maintain its information content. It represents the desired state of the asset based on that process, and is compared with the actual current state for compliance validation.
Discovery Library	Discovery Library technologies consist of a specification, a set of components, and best practices for communicating the discovery of resources and the relationships among resources within the enterprise. The Discovery Library XML schema specification, called Identity Markup Language (IdML), is used to specify resources. Books within the Discovery Library are XML files that contain discovery information, identity of resources, and resource relationships. The set of Discovery Library components comprise code (written by IBM and other companies) that extracts data and transforms it into the XML specification.
Identity Markup Language (IdML)	The Discovery Library XML schema is called IdML. An extension of the CDM XML schema developed by IBM Tivoli Software, IdML describes a set of valid operations to act on CDM-based resource instances. This easy-to-read XML format provides details about resources and the relationships between resources, as well as particular operations, such as create, modify, and delete, that act against the resource data.

Term/concept	Definition
Discovery Library Adapters (DLAs)	Discovery Library Adapters (DLAs) are the workers in the Discovery Library. DLAs exploit mechanisms that are native to sources of discovered information, such as the TMTP transaction topology, the WBI Modeler business process topology, and so on, to extract specific information about resources and, in particular, resource relationships. The DLAs are code, written in any programming language, that transforms information into a file of IdML-schema-compliant data that readers process for domain-specific purposes. The end goal of the information from the adapters is to *discover* and keep current the sets of resources and relationships that comprise business applications, support IT processes or business processes, and so on. DLAs are command-line executable.

CDM overview

The Common Data Model is a consistent, integrated, logical data model that defines the general characteristics of information stored in the CMDB. The model specifies how this data is organized to correspond to real-world entities and defines the relationships between the entities. The CDM represents management information in a way that is easy for consuming management applications to use.

CDM components

Based on the Unified Modeling Language (UML), the CDM represents management information in terms of entities (called ManagedElements) and the relationships among those entities. The CDM strives to include information from all of the logical models in use (such as CIM, BPEL, ITIL, SNA, and TMf) and integrates them into a single consistent model.

The CDM and associated documents can be viewed at the Tivoli CDM Web site, which is included on the Tivoli Application Dependency Discovery Manager (TADDM) product DVD. To access the Web site, unzip the following file, which is located in the TADDM installation directory:

```
install_root/cmdb/dist/sdk/doc/model/CDMWebsite.zip
```

Note that while all configuration items are stored in the CMDB and are modeled by the CDM, we expect the CMDB and the CDM to include other information that is not directly represented as CIs.

Concepts

The CDM draws most of its concepts from UML, and the contents of the model can be used in UML development tools, such as IBM Rational® Software Architect.

Attributes

At the most basic level of granularity, the CDM represents atomic data as an attribute, as defined by UML. An attribute has an associated data type, a possible default value, and a specification of whether the attribute is single-valued or multi-valued. Certain data types, and enumerations, limit the actual values that an attribute can contain. All attributes in the CDM are globally defined, which means that an attribute with the same name has the same meaning, regardless of the context in which it is used. This is to foster consistent definition and use of the attribute in various environments and circumstances, such as events. Some examples of attributes are:

- Manufacturer
- MemorySize
- PrimaryOwner
- PrimaryMacAddress

Classes

Attributes are grouped within the CDM into entities that correspond to items in the real world, such as computers, users, or business processes. This grouping of attributes is called a *class*. Classes in the CDM are arranged into a single-inheritance hierarchy that enables attributes to be shared among classes.

In some cases, classes are *abstract*. Abstract classes contain common characteristics of entities, but instances of these classes cannot be created. ModelObject, the root of the class hierarchy, is an example of an abstract class. A vast majority of the classes in the CDM are *concrete*, which means that instances of them can be created in the CMDB.

Note that the class hierarchy of the CDM is rooted in the class ModelObject, not ConfigurationItem. In addition to CIs, other kinds of data will be stored in the CMDB and modeled using the CDM.

Interfaces

Many situations that commonly occur in the real world lead people to use multiple inheritance, which is supported by UML but not by the CDM. In order to handle these situations, the CDM includes the concept of an interface, which is a consistent collection of attributes (or a consistent source or target of a relationship) that can be *included* in a class definition anywhere in the class hierarchy. This is similar to the way in which Java™ handles interfaces, except that the CDM includes only data, not methods.

Interfaces themselves can be derived from other interfaces, thus forming another inheritance hierarchy. However, while an interface hierarchy can have multiple roots, the derivation hierarchy cannot mix interfaces and classes. Classes can be derived only from other classes, and interfaces can be derived only from other interfaces.

Relationships

One of the most important purposes of the CMDB is to store relationships between entities in the real world. The CDM therefore places a lot of focus on the definition of relationships between classes and interfaces, and assigns a specific semantic meaning to the relationship. For example, a relationship called "runsOn" may represent the fact that a piece of software executes in a particular environment.

Relationships in the CDM are related to, but differ from, a similar concept in UML called *associations*. An association is a semantic link between classes in UML. An example is a *realization*, where one entity makes a particular interface available. Nothing in UML forces a user to express the meaning of an association. You can simply draw a line between two entities. In the CDM, all associations (other than generalization and realization) are named or typed. The name of the association gives it a corresponding meaning, therefore making the association a relationship. All associations with the same name have the same meaning.

Naming and identification

In addition to representing and storing relationships between entities, the CMDB provides a correlation mechanism between entities. For example, two management products might discover a single computer system and call them different names. It is important to represent this as a single entity. In order to foster consistent identification of entities in the CMDB, the CDM formally defines the ways in which each type of entity (each class) is identified. To do so, the model uses *naming rules*.

Naming rules list the attributes that provide identifying characteristics, the combination of attributes are that needed to identify the entity, and the context that makes the identification unique. Two examples of naming rules are:

- Combining "Manufacturer," "MachineType," "Model," and "SerialNumber" gives a unique identification of a computer.
- The "DriveLetter" of a logical disk gives a unique identification of the disk within the context of an operating system.

Correlation in the CMDB is fostered by a consistent use of these rules and an understanding of which rules identify instances of the same type. When multiple names for the same instance arise, they are called *aliases*, and the CMDB represents the duplicates as a single instance.

Consistent formation of names using the naming rules also allows the CMDB (or applications) to generate useful binary tokens known as globally unique identifiers (GUIDs) for the instances.

Example template scenarios

This section provides a set of example template scenarios that help you understand the Common Data Model. For additional details about the classes, or for more information about classes that are not covered in these scenarios, refer to the CDM Web site, which is included on the IBM Tivoli Application Dependency Discovery Manager (TADDM) product DVD. To access the Web site, unzip the following file, which is located in the TADDM installation directory:

`install_root/cmdb/dist/sdk/doc/model/CDMWebsite.zip`

Each scenario description provided in this section includes the features described below:.

An instance diagram

For each scenario, the instance diagram shows all of the classes and relationships that are used to model the specific scenario. For the sake of clarity, the following guidelines are observed:

- If more than one instance of a class is required, it is shown as a stack of boxes. For example, in the Multi-Processor Linux/Unix Operating System Server scenario, multiple CPUs are shown as a stack of boxes.
- If a scenario involves multiple servers, multiple ComputerSystem boxes are shown. Each ComputerSystem box represents a separate server.
- If a scenario refers to another scenario, the details of the referred-to scenario are not repeated. For example, since the details of a WebSphere® server are covered in the IBM WebSphere single-node server scenario, in a J2EE™ transaction scenario, we only show that an activity in that transaction uses a WebSphere server. No details of the WebSphere server are repeated in the J2EE transaction.

Naming rules and naming attributes

CDM defines naming rules and uses them to foster consistent identification of resources in the CMDB. These rules formally define the ways in which each type of resource (each class) is identified. The rules list the attributes that provide identifying characteristics, such as the combination of attributes needed to uniquely identify the resource. When there is not enough information to uniquely identify a resource through attributes, the CDM uses a reference to another resource instance to provide for a naming context.

For example, an instance of OperatingSystem uses the attribute OSName. Given that OSName represents the brand name of the operating system, this attribute alone does not make the OperatingSystem instance unique. In order to uniquely identify the operating system, the CDM provides a naming context (referred to as *superior*) for this naming rule, requiring the ComputerSystem instance that the OperatingSystem instance is installedOn. Therefore, naming an OperatingSystem resource requires a relationship (in this example, the installedOn relationship) along with the attribute OSName.

When a resource is created, if insufficient naming attributes are provided to derive a valid unique name for the resource, the create request is rejected.

Table 2 illustrates the naming rules for the OperatingSystem class.

Table 2 Naming rules for the OperatingSystem class

Class (IdML element name)	**Naming rules**	**Superior (or naming context)**
OperatingSystem* (sys.OperatingSystem)	0=”superior, Name” 1=”superior, OsId” 2=”systemGuid” 3=”superior, OSName” 4=”managedSystemName” 5=”FQDN”	An instance representing the ComputerSystem the OperatingSystem is installedOn

As the table shows, creating an OperatingSystem requires that at least one of the following combinations of attributes is provided:

- An instance representing the Computer System the OperatingSystem is *installedOn* and Name
- A instance representing the Computer System the OperatingSystem is *installedOn* and Operating System ID
- systemGUID
- A instance representing the Computer System the OperatingSystem is *installedOn* and OSName

- ManagedSystemName
- Fully qualified host name (FQDN)

In addition to adhering to the CDM naming rules and naming attributes, the scenarios presented in this section also provide the following clarifications:

- An asterisk (*) appended to the end of a class name indicates that the class is a configuration item.
- The IdML element name of the class used to load a resource of the class type via a Discovery Library Adapter (DLA) is provided. The IdML element name represents the name of the CDM class as an XML element tag in the IdML space.

In the OperatingSystem example, an OperatingSystem is a CI, and the IdML element name of the OperatingSystem class is sys.OperatingSystem. For examples of other IdML elements, refer to “Sample model objects” on page 106.

Tables of important relationships

For each scenario, a table lists all of the important relationships, including the source, the target, the relationship type, and the cardinality of the relationship, such as one to many (1:m), many to one (m:1), many to many (m:n), or one to one (1:1). Table 3 illustrates several examples of important relationships.

Table 3 Examples of important relationships

Source class	Relationship type	Target class	Cardinality
ComputerSystem	contains	IpInterface	1:m
IpInterface	bindsTo	IpV4Address	1:m
Fqdn	assignedTo	IpV4Address	1:1
WindowsOperatingSystem	runsOn	ComputerSystem	1:1
WindowsOperatingSystem	installedOn	ComputerSystem	m:1
SoftwareInstallation	installedOn	WindowsOperatingSystem	m:1

Table 3 on page 9 shows the following about important relationships:

- A computer system can contain one or more IP interfaces.
- An IP interface can bind to one or more IP V4 Addresses.
- A fully qualified host name is assigned to one and only one IP V4 Address.
- Multiple Windows® operating systems can be installed on one computer, but only one can be running.
- Multiple software installations can be installed on a Windows operating system.

Potential questions about the scenario

For each scenario, this section contains a list of potential questions that the model of a specific scenario will need to be able to answer. For example, one of the potential questions for a standard Windows operating system server is: What operating system is on this computer? To answer this question, the model of the scenario needs to include both the ComputerSystem and the OperatingSystem classes, and the relationship between these two classes. Therefore, it is very important to have the correct set of potential questions to validate the completeness of the model of a scenario.

Operating system server scenarios

This section presents CDM scenarios for five operating system servers: the Microsoft® Windows, Multi-process Unix/Linux®, zSeries®, Virtualization (VMware), and Virtualization (Multi-LPAR zSeries) servers.

Microsoft Windows server

This scenario shows a Microsoft Windows server that has a Microsoft Windows operating system installed and running on it. The operating system is booted from a local file system. The server also has multiple software installations installed on it. Only one software installation is shown in the diagram: DB2® V8.2. However, the server can have other software installations, such as Microsoft IE 6.0.2, Lotus® Notes® Client 7, Microsoft Office Visio® Professional 2007, and so on. This server also has a V4 IP address and a fully qualified host name. The IP V4 address shown in the diagram is an example. IP V6 is supported by replacing the use of the IpV4Address class with the IpV6Address class.

Naming rules and naming attributes

See "Naming rules and naming attributes " on page 78 for the naming rules and naming attributes of each class shown in Figure 1.

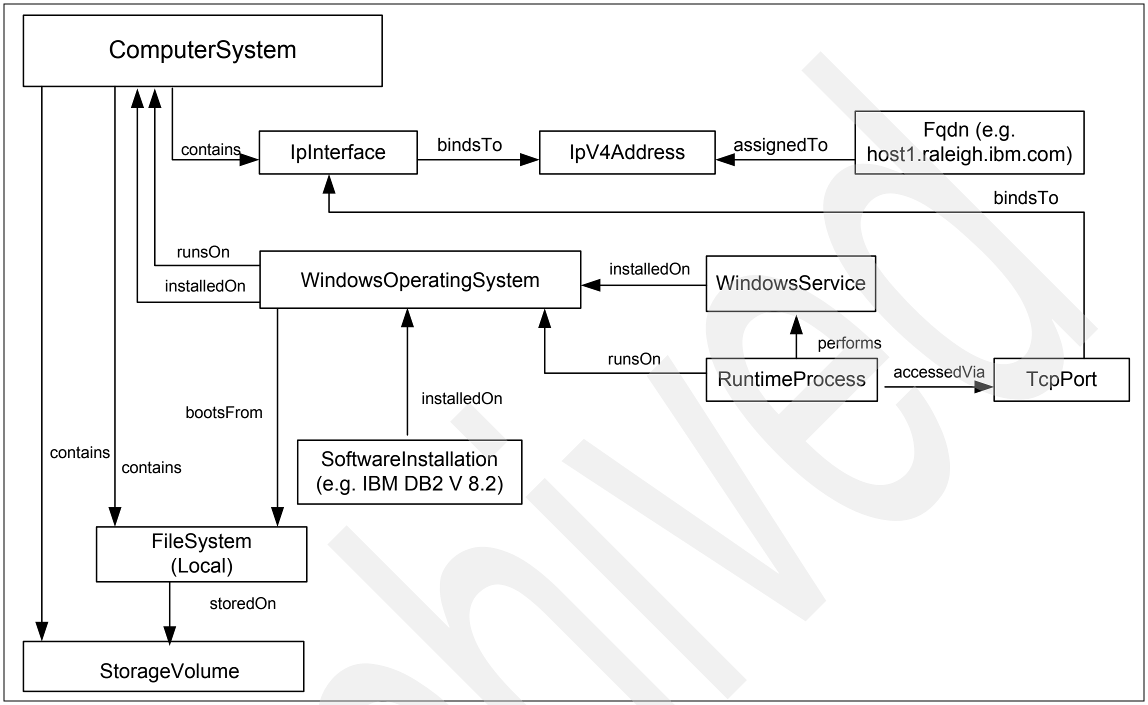

Figure 1 Instance diagram, Microsoft Windows server

Important relationships

Table 4 shows the important relationships in this scenario.

Table 4 Important relationships, Microsoft Windows server

Source class	Relationship type	Target class	Cardinality
ComputerSystem	contains	Filesystem	1:m
ComputerSystem	contains	IpInterface	1:m
ComputerSystem	contains	StorageVolume	1:m
Filesystem	storedOn	StorageVolume	m:1
Fqdn	assignedTo	IpV4Address	1:1
IpInterface	bindsTo	IpV4Address	1:m

Source class	Relationship type	Target class	Cardinality
RuntimeProcess	accessedVia	TcpPort	1:m
RuntimeProcess	performs	WindowsService	m:n
RuntimeProcess	runsOn	WindowsOperatingSystem	m:1
SoftwareInstallation	installedOn	WindowsOperatingSystem	m:1
TcpPort	bindsTo	IpInterface	m:1
WindowsOperatingSystem	bootsFrom	Filesystem	1:1
WindowsOperatingSystem	installedOn	ComputerSystem	m:1
WindowsOperatingSystem	runsOn	ComputerSystem	1:1
WindowsService	installedOn	WindowsOperatingSystem	m:1

Potential questions

Here are some related questions:

- What operating system is on this computer?
- What software is installed on this computer?
- How much free disk space is on this computer?
- How much free file system space (or available space) is on this computer?
- Which computers have a personal firewall installed?
- Which computers do not have firewall or antivirus software installed?
- Which computers have a back-level version of the anti-virus software?

Multi-processor UNIX/Linux server

This scenario shows a multi-processor UNIX/Linux server that has a UNIX® operating system installed and running on it. The server contains a local file system and has multiple software installations installed on it. Only one software installation is shown in the diagram: IBM Lotus Notes Client. However, it can have other software installations, such as IBM DB2, Firewall, LDAP, and so on. This server also has a V4 IP address and a fully qualified host name. The IP V4 address shown in Figure 2 on page 13 is an example. IP V6 is supported by replacing the use of the IpV4Address class with the IpV6Address class.

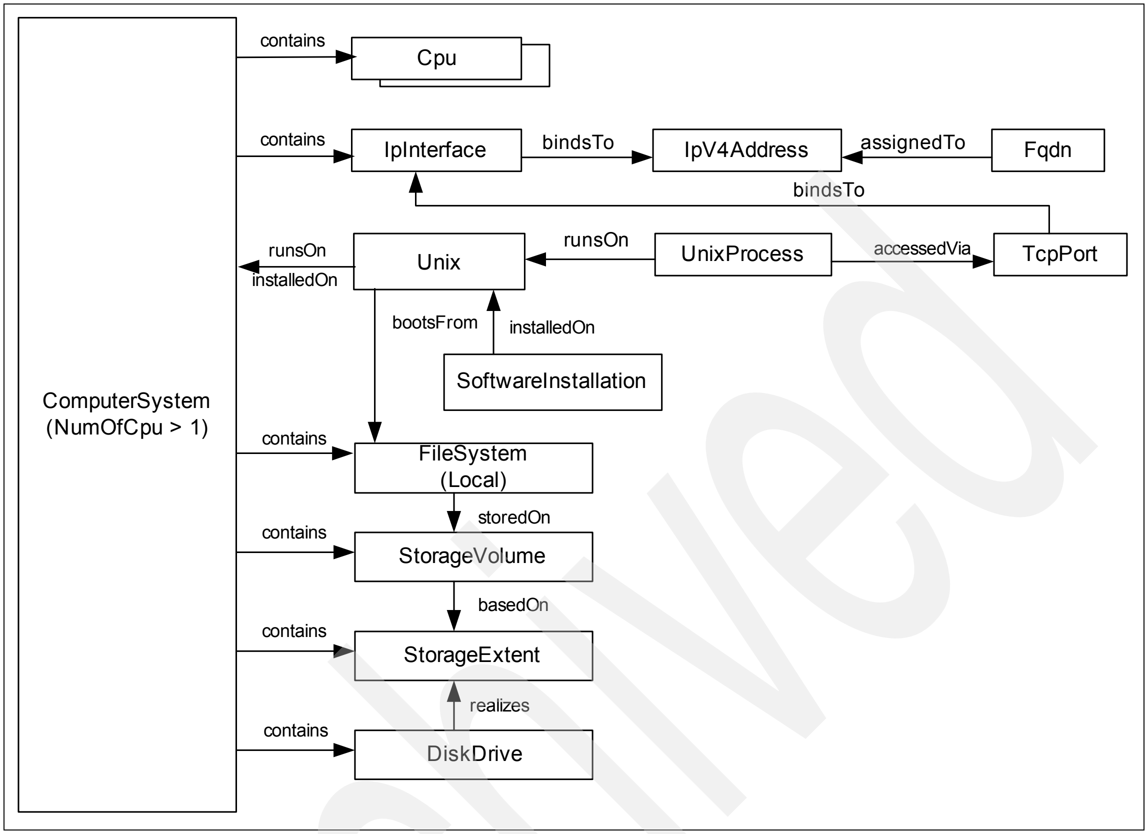

Figure 2 Instance diagram, multi-process UNIX/Linux server

Naming rules and naming attributes

See “Naming rules and naming attributes ” on page 78 for the naming rules and naming attributes of each class shown in Figure 2.

Important relationships

Table 5 shows the important relationships in this scenario.

Table 5 Important relationships, multi-processor UNIX/Linux server

Source class	Relationship type	Target class	Cardinality
ComputerSystem	contains	Cpu	1:m
ComputerSystem	contains	DiskDrive	1:m
ComputerSystem	contains	Filesystem	1:m
ComputerSystem	contains	IpInterface	1:m
ComputerSystem	contains	StorageVolume	1:m
ComputerSystem	contains	StorageExtent	1:m
DiskDrive	realizes	StorageExtent	m:n
Filesystem	storedOn	StorageVolume	m:1
Fqdn	assignedTo	IpV4Address	1:1
IpInterface	bindsTo	IpV4Address	1:m
SoftwareInstallation	installedOn	Unix	m:1
StorageVolume	basedOn	StorageExtent	m:1
TcpPort	bindsTo	IpInterface	m:1
Unix	bootsFrom	Filesystem	m:1
Unix	installedOn	ComputerSystem	m:1
Unix	runsOn	ComputerSystem	1:1
UnixProcess	accessedVia	TcpPort	1:m
UnixProcess	runsOn	Unix	m:1

Usage or implementation notes

The attributes of some of the classes must be set as follows: The numCPUs attribute of this server should be set to greater than 1, and details of each CPU are provided via the CPU model object contained in the ComputerSystem model object.

Potential questions

Here are some related questions:

- What operating system is on this computer?
- What software is installed on this computer?
- How much free disk space is on this computer?
- How much free file system space (or available space) is on this computer?
- How much free formatted disk space is on this computer? How much unformatted disk space is on this computer?
- Which computers support this test environment (for example, software version)?
- Which computers support this test configuration (for example, computer model, CPU speed, memory size, and so on)?
- Which servers are currently running Apache?
- What is the bandwidth (or speed) of their network connectivity?
- If I take the volume off-line, what systems are impacted?
- Are servers using these critical files impacted if I take this volume off-line?
- How many processors are on this computer?
- What is the average CPU speed of all these processors?

zSeries server

This scenario shows a zSeries server that has a zOS operating system running on it. This server supports multiple subsystems (such as DB2, IMS™, and MQ), a CICS® region, and a WebSphere Server. The details of the DB2 subsystem and the WebSphereServer can be found in “DB2 (zOS)” on page 38 and “IBM WebSphere single-node server” on page 23, respectively. Again, the IP V4 address shown in the diagram is an example. IP V6 is supported by replacing the use of the IpV4Address class with the IpV6Address class.

Naming rules and naming attributes

See for the naming rules and naming attributes of each class shown in Figure 3.

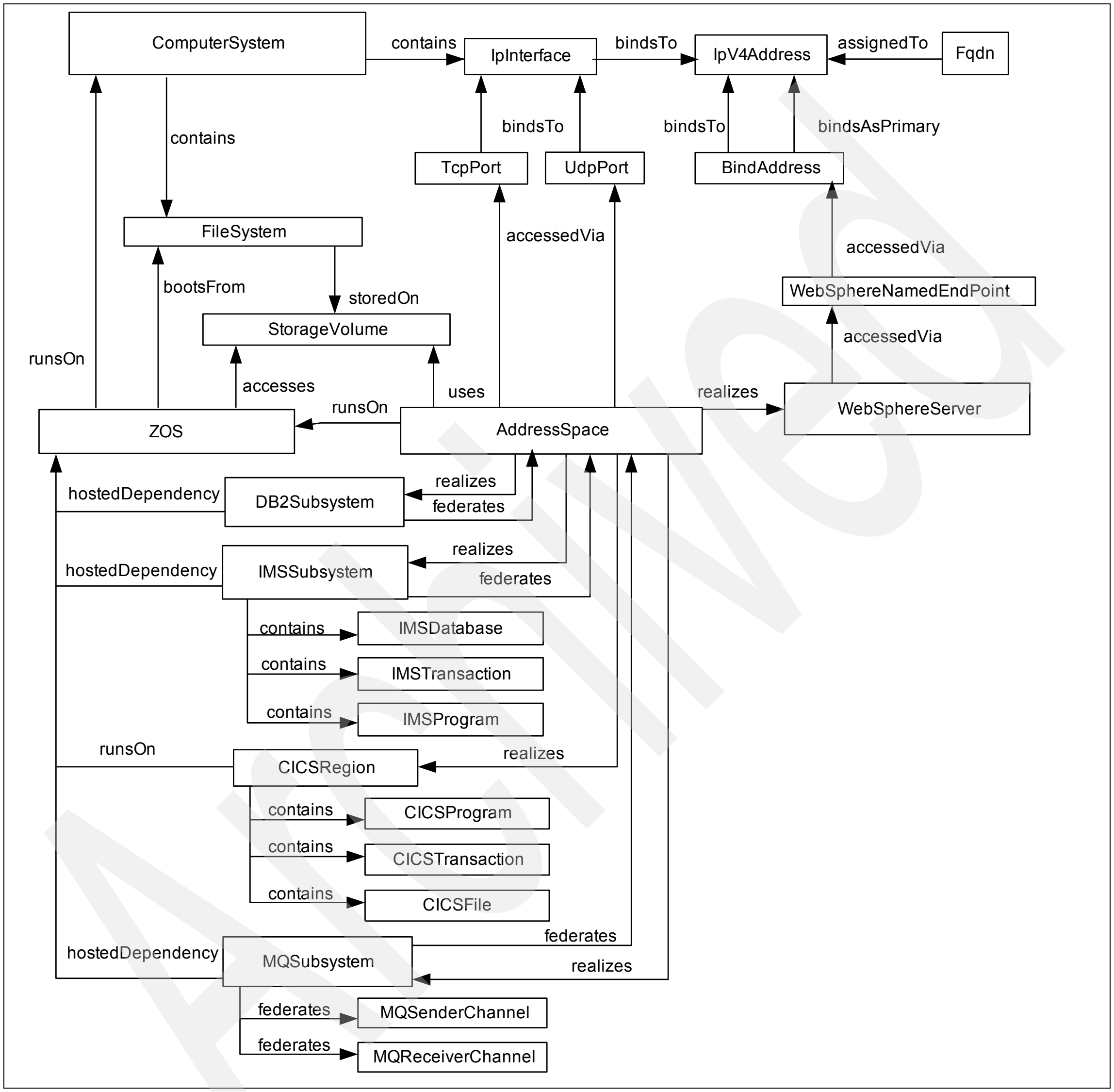

Figure 3 Instance diagram, zSeries server

Important relationships

Table 6 shows the important relationships in this scenario.

Table 6 Important relationships, zSeries server

Source class	Relationship type	Target class	Cardinality
AddressSpace	accessedVia	TcpPort	1:m
AddressSpace	accessedVia	UdpPort	1:m
AddressSpace	realizes	CICSRegion	m:1
AddressSpace	realizes	Db2Subsystem	m:1
AddressSpace	realizes	IMSSubsystem	m:1
AddressSpace	realizes	MQSubsystem	m:1
AddressSpace	realizes	WebSphereServer	1:1
AddressSpace	runsOn	ZOS	m:1
AddressSpace	uses	StorageVolume	m:n
BindAddress	bindsAsPrimary	IpV4Address	1:1
BindAddress	bindsTo	IpV4Address	1:1
CICSRegion	contains	CICSFile	1:m
CICSRegion	contains	CICSProgram	1:m
CICSRegion	contains	CICSTransaction	1:m
CICSRegion	runsOn	ZOS	m:1
ComputerSystem	contains	FileSystem	1:m
ComputerSystem	contains	IpInterface	1:m
Db2Subsystem	federates	AddressSpace	m:1
Db2Subsystem	hostedDependency	ZOS	m:1
FileSystem	storedOn	StorageVolume	m:1
Fqdn	assignedTo	IpV4Address	1:1
IMSSubsystem	contains	IMSDatabase	1:m

Source class	Relationship type	Target class	Cardinality
IMSSubsystem	contains	IMSProgram	1:m
IMSSubsystem	contains	IMSTransactio n	1:m
IMSSubsystem	federates	AddressSpace	m:1
IMSSubsystem	hostedDepende ncy	ZOS	m:1
IpInterface	bindsTo	IpV4Address	1:m
MQSubsystem	federates	AddressSpace	m:1
MQSubsystem	federates	MQReceiverC hannel	1:m
MQSubsystem	federates	MQSenderCh annel	1:m
MQSubsystem	hostedDepende ncy	ZOS	m:1
TcpPort	bindsTo	IpInterface	m:1
UdpPort	bindsTo	IpInterface	m:1
WebSphereNamedEnd Point	accessedVia	bindAddress	1:1
WebSphereServer	accessedVia	WebSphereNa medEndPoint	1:m
ZOS	accesses	StorageVolum e	m:n
ZOS	bootsFrom	FileSystem	m:1
ZOS	runsOn	ComputerSyst em	1:1

Potential questions

Some related questions are:

- What operating system is on this computer?
- What software or subsystem is installed on this computer?
- How much free disk space is on this computer?
- How much free file system space (or available space) is on this computer?
- Is WebSphere currently running on the zOS instance?

- What is the bandwidth (or speed) of their network connectivity?
- If I take a volume off-line, what systems are impacted?
- If I take a volume off-line, what middleware applications are impacted?
- Are servers using these critical files impacted if I take this volume off-line?

Virtualization (VMware) server

This scenario shows a Microsoft Windows operating system server that is partitioned into two VMware images. The first VMware hosts a Windows virtual machine with a DB2 Server running on it, while the second VMware hosts a Linux virtual machine with an Oracle® Server running on it.

Naming rules and naming attributes

See "Naming rules and naming attributes" on page 8 for the naming rules and naming attributes of each class shown in Figure 4.

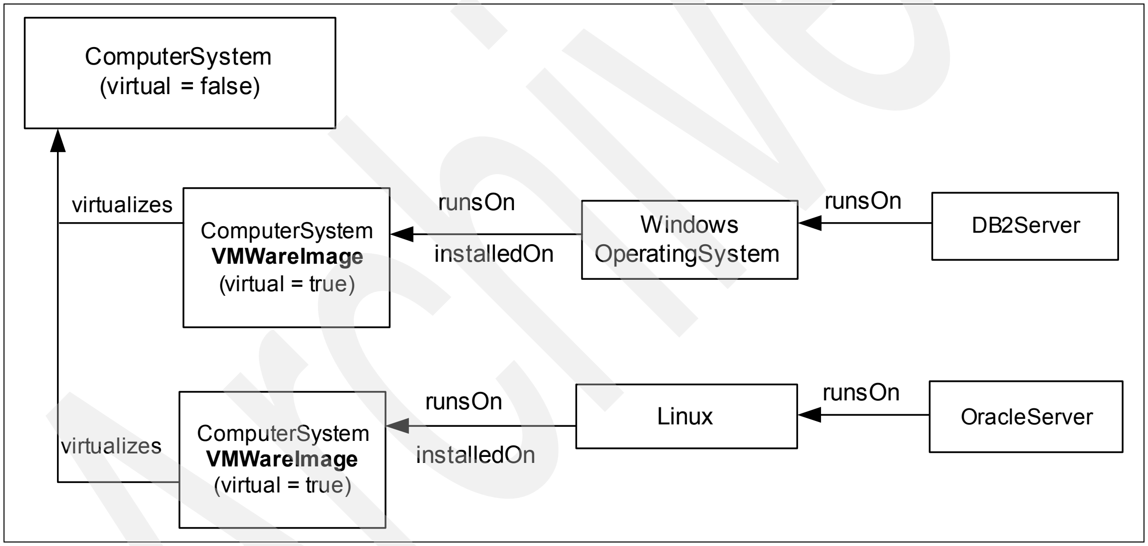

Figure 4 Instance diagram, virtualization (VMware) server

Important relationships

Table 7 on page 20 shows the important relationships in this scenario.

Table 7 Important relationships, virtualization (VMware) server

Source class	Relationship type	Target class	Cardinality
ComputerSystem	virtualizes	ComputerSystem	m:1
Db2Server	runsOn	WindowsOperatingSystem	m:1
Linux	installedOn	ComputerSystem	m:1
Linux	runsOn	ComputerSystem	1:1
OracleServer	runsOn	Linux	m:1
WindowsOperating System	installedOn	ComputerSystem	m:1
WindowsOperating System	runsOn	ComputerSystem	1:1

Usage or implementation notes

If a ComputerSystem is a VMwareImage, the Virtual attribute must be set to true, the VMID attribute must be set to the VM Ware image GUID, and the IsVMIDanLPAR attribute must be set to false.

Potential questions

Here are some related questions:

- What virtual machines are hosted on this computer?
- What operating systems are hosted on this computer?
- What software is installed on this computer?

Virtualization (multi-LPAR zSeries) Server

This scenario shows a zSeries operating system server that is partitioned into two LPARs. The first LPAR hosts a zOS, while the second LPAR hosts two virtual machines: one virtual machine hosts zOS, the other hosts zLinux.

Naming rules and naming attributes

See for naming rules and naming attributes of each class shown in Figure 5.

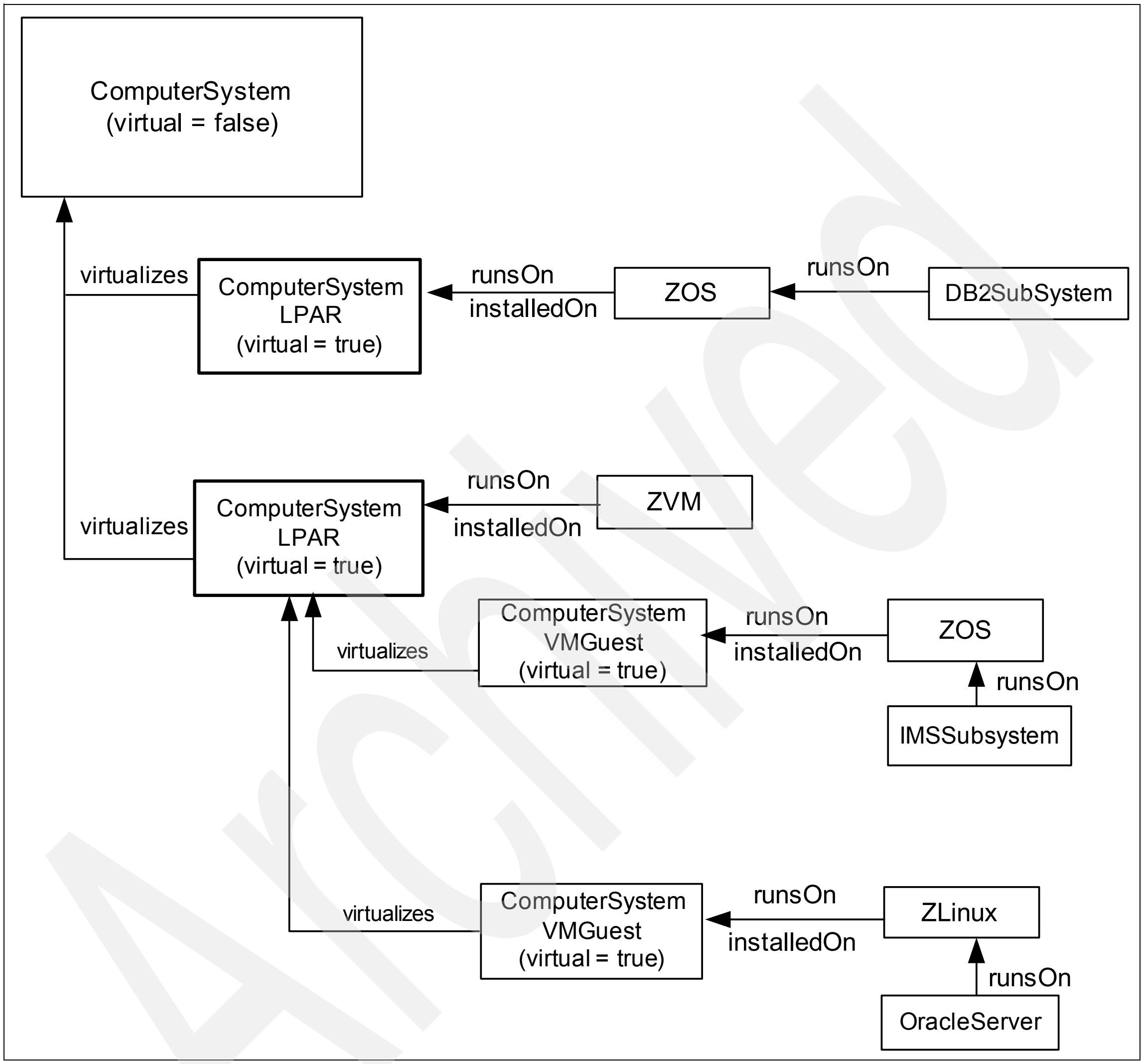

Figure 5   Instance diagram, virtualization (multi-LPAR) server

Important relationships

Table 8 shows the important relationships in this scenario.

Table 8 .Important relationships, virtualization (multi-LPAR zSeries) server

Source class	Relationship type	Relationship type	Cardinality
ComputerSystem	virtualizes	ComputerSystem	m:1
Db2Subsystem	runsOn	ZOS	m:1
IMSSubsystem	runsOn	ZOS	m:1
OracleServer	runsOn	ZLinux	m:1
ZLinux	installedOn	ComputerSystem	m:1
ZLinux	runsOn	ComputerSystem	1:1
ZVM	installedOn	ComputerSystem	m:1
ZVM	runsOn	ComputerSystem	1:1
ZOS	installedOn	ComputerSystem	m:1
ZOS	runsOn	ComputerSystem	m:1

Usage or implementation notes

If a ComputerSystem is an LPAR, the Virtual attribute must be set to true, the VMID attribute must be set to the LPAR ID, and IsVMIDanLPAR must be set to true.

If a ComputerSystem is a VM guest, the Virtual attribute must be set to true, the VMID attribute must be set to the VM guest ID, and IsVMIDanLPAR must be set to false.

Potential questions

Here are some related questions:

- What virtual machines are hosted on this computer?
- What operating systems are hosted on this virtual machine?
- What software is installed on this virtual machine?

Web/application server scenarios

This section presents CDM scenarios for four Web/application servers: the IBM WebSphere Single-Node server, the IBM WebSphere Multi-Node server, the WebLogic Single-Node server, and the WebLogic Multi-Node server.

IBM WebSphere single-node server

This scenario shows an IBM WebSphere server that is a stand-alone server, not participating in a cell or cluster. J2EE applications are deployed to the server. The server contains an EJB™ container and a Web container for running these applications. The server uses a JDBC™ data source to connect to databases, and uses a JMS server to connect to a messaging system. The server also provides endpoints for accessing Web services. The server runs on an operating system. If the server is running in a zOS environment, it is controlled by an address space, and it can use a CICSRegion and various zOS Subsystems.

Naming rules and naming attributes

See for the naming rules and naming attributes of each class shown in Figure 6.

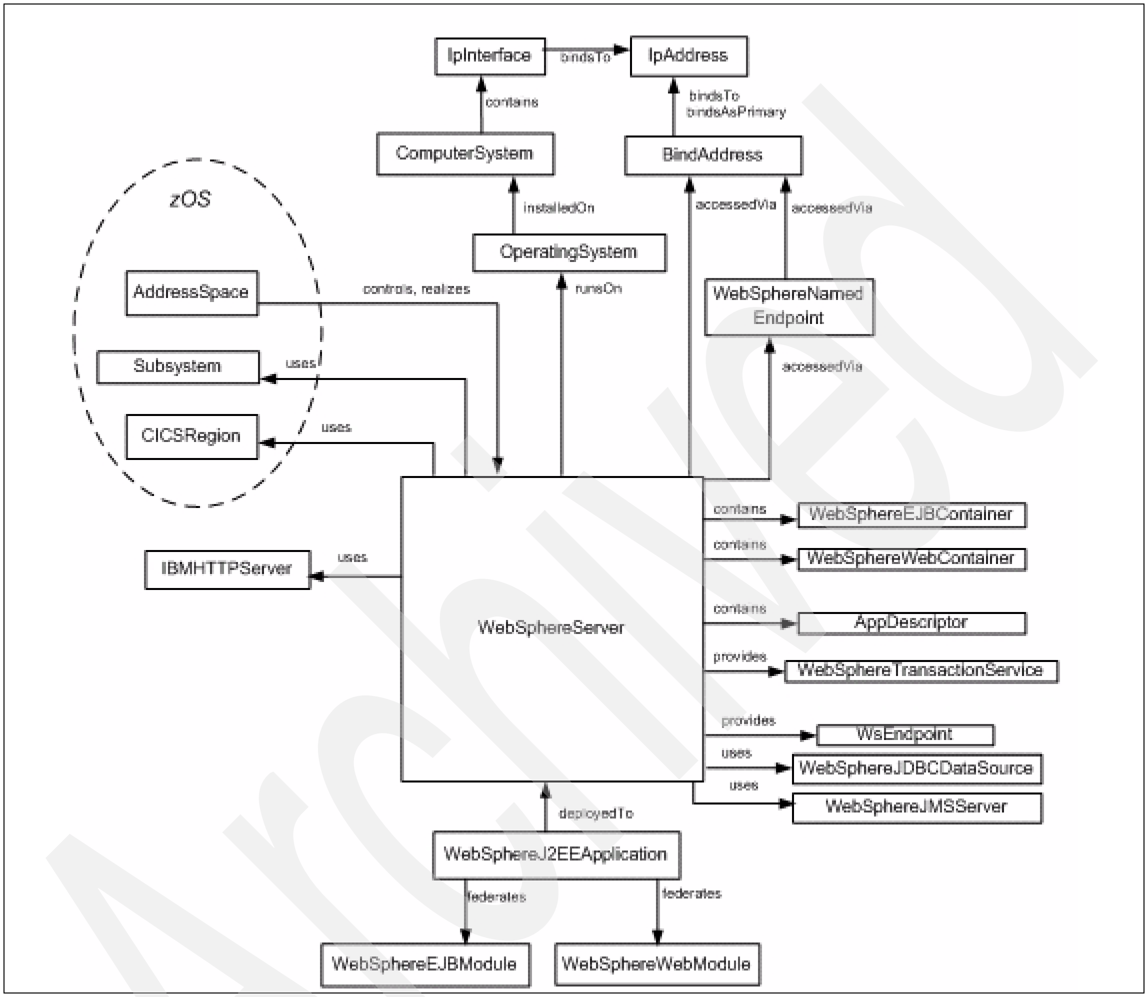

Figure 6 Instance diagram, IBM WebSphere single-node server

Important relationships

Table 9 shows the important relationships in this scenario.

Table 9 Important relationships, IBM WebSphere single-node server

Source class	Relationship type	Target class	Cardinality
AddressSpace	controls	WebSphereServer	1:1

Source class	Relationship type	Target class	Cardinality
AddressSpace	realizes	WebSphereServer	1:1
BindAddress	bindsAsPrimary	IpAddress	1:1
BindAddress	bindsTo	IpAddress	1:1
ComputerSystem	contains	IpInterface	1:m
IpInterface	bindsTo	IpAddress	1:m
OperatingSystem	installedOn	ComputerSystem	m:1
WebSphereJ2EE Application	deployedTo	WebSphereServer	m:n
WebSphereJ2EE Application	federates	WebSphereEJBModule	m:n
WebSphereJ2EE Application	federates	WebSphereWebModule	m:n
WebSphereName dEndpoint	accessedVia	BindAddress	1:1
WebSphereServer	accessedVia	BindAddress	1:1
WebSphereServer	accessedVia	WebSphereNamedEnd point	1:m
WebSphereServer	contains	AppDescriptor	1:m
WebSphereServer	contains	WebSphereEJBContain er	1:m
WebSphereServer	contains	WebSphereWebContai ner	1:m
WebSphereServer	provides	WebSphereTransaction Service	1:m
WebSphereServer	provides	WSEndpoint	1:m
WebSphereServer	runsOn	OperatingSystem	m:1
WebSphereServer	uses	IBMHTTPServer	m:n
WebSphereServer	uses	WebSphereJDBCDataS ource	1:m
WebSphereServer	uses	WebSphereJMSServer	1:m

Source class	Relationship type	Target class	Cardinality
WebSphereServer (zOS only)	uses	CICSRegion	m:1
WebSphereServer (zOS only)	uses	Subsystem	1:m

Potential questions

Here are some related questions:

- On what IP address and port is the WebSphere server listening?
- What J2EE applications are deployed to this server?
- What EJB modules and Web modules make up these applications?
- On what operating system and computer system is the WebSphere server running?
- What Web service endpoints are provided by this server, and what is the IP address and port for each Web Service endpoint?
- What are the JNDI addresses for the data sources and JMS servers used by the WebSphere server?
- What JDBC driver (implementation class name) is being used for each datasource?
- What databases are being accessed by each datasource?
- (zOS only) Which zOS subsystems are being used by the WebSphere server?

IBM WebSphere multi-node server

This scenario shows an IBM WebSphere server that is a part of a cluster of servers. The server is a member of the cluster, and a cell federates a cluster. A node is a member of a cell and is managed by a node agent. A cell is configured using a global security settings object, and is managed by a deployment manager. The cell can access the user registry via its global security settings object. J2EE applications are deployed to either the server or to the cell. Otherwise, the description of the WebSphere single-node server in "IBM WebSphere single-node server" on page 23 applies.

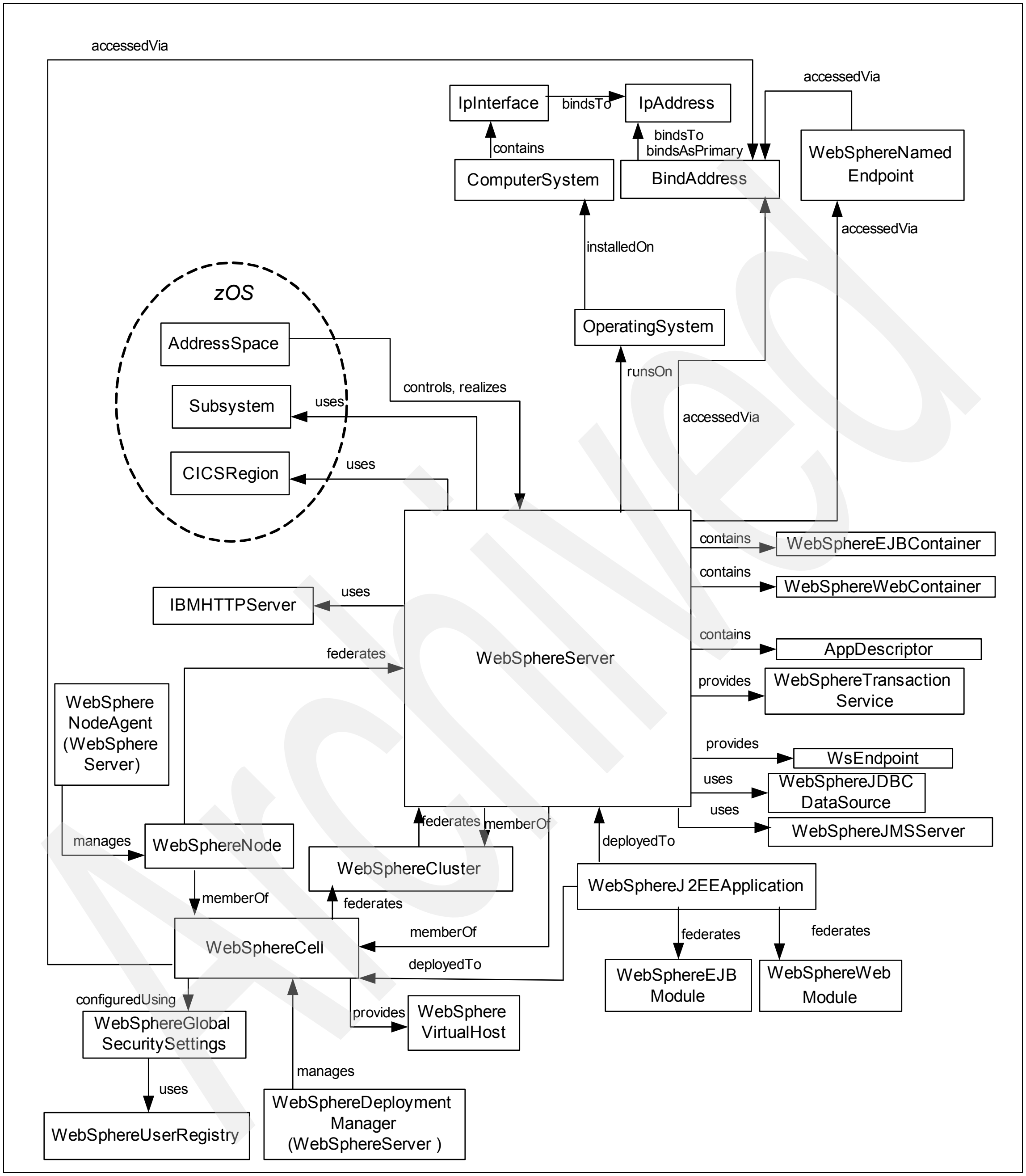

Figure 7 Instance diagram, IBM WebSphere multi-node server

Naming rules and naming attributes

See for the naming rules and naming attributes of each class shown in Figure 7 on page 27.

Important relationships

Table 10 shows the important relationships in this scenario. This list of important relationships is in addition to all of the relationships listed in "IBM WebSphere single-node server" on page 23.

Table 10 Important relationships, WebSphere multi-node server

Source class	Relationship type	Target class	Cardinality
WebSphereCell	accessedVia	BindAddress	1:1
WebSphereCell	configuredUsing	WebSphereGlobalSecuritySettings	1:1
WebSphereCell	federates	WebSphereCluster	1:m
WebSphereCell	provides	WebSphereVirtualHost	1:m
WebSphereCluster	federates	WebSphereServer	1:m
WebSphereDeploymentManager	manages	WebSphereCell	1:1
WebSphereGlobalSecuritySettings	uses	WebSphereUserRegistry	1:1
WebSphereJ2EEApplication	deployedTo	WebSphereCell	m:1
WebSphereNode	federates	WebSphereServer	1:m
WebSphereNode	memberOf	WebSphereCell	m:1
WebSphereNodeAgent	manages	WebSphereNode	1:1
WebSphereServer	memberOf	WebSphereCell	m:1
WebSphereServer	memberOf	WebSphereCluster	m:1

Potential questions

These queries are in addition to all of the queries listed in “IBM WebSphere single-node server” on page 23.

- Which cluster and cell is the WebSphere server a member of?
- Which applications are deployed to the cell?
- Is global security enabled for the cell?
- Which user registry is being used by the cell (LocalOS, LDAP, other)?
- What other WebSphere servers are running on the same node?
- What other WebSphere servers are parts of the same cell?

WebLogic single-node server

This scenario shows a WebLogic server that is a stand-alone server, not participating in a cluster. J2EE applications are deployed to the server. The server contains an EJB container and a Web container for running these applications. The server uses a JDBC data source to connect to databases, and uses a JMS server to connect to a messaging system. The server contains application descriptors and provides endpoints for accessing Web services. The server runs on an operating system.

Naming rules and naming attributes

See for the naming rules and naming attributes of each class shown in Figure 8.

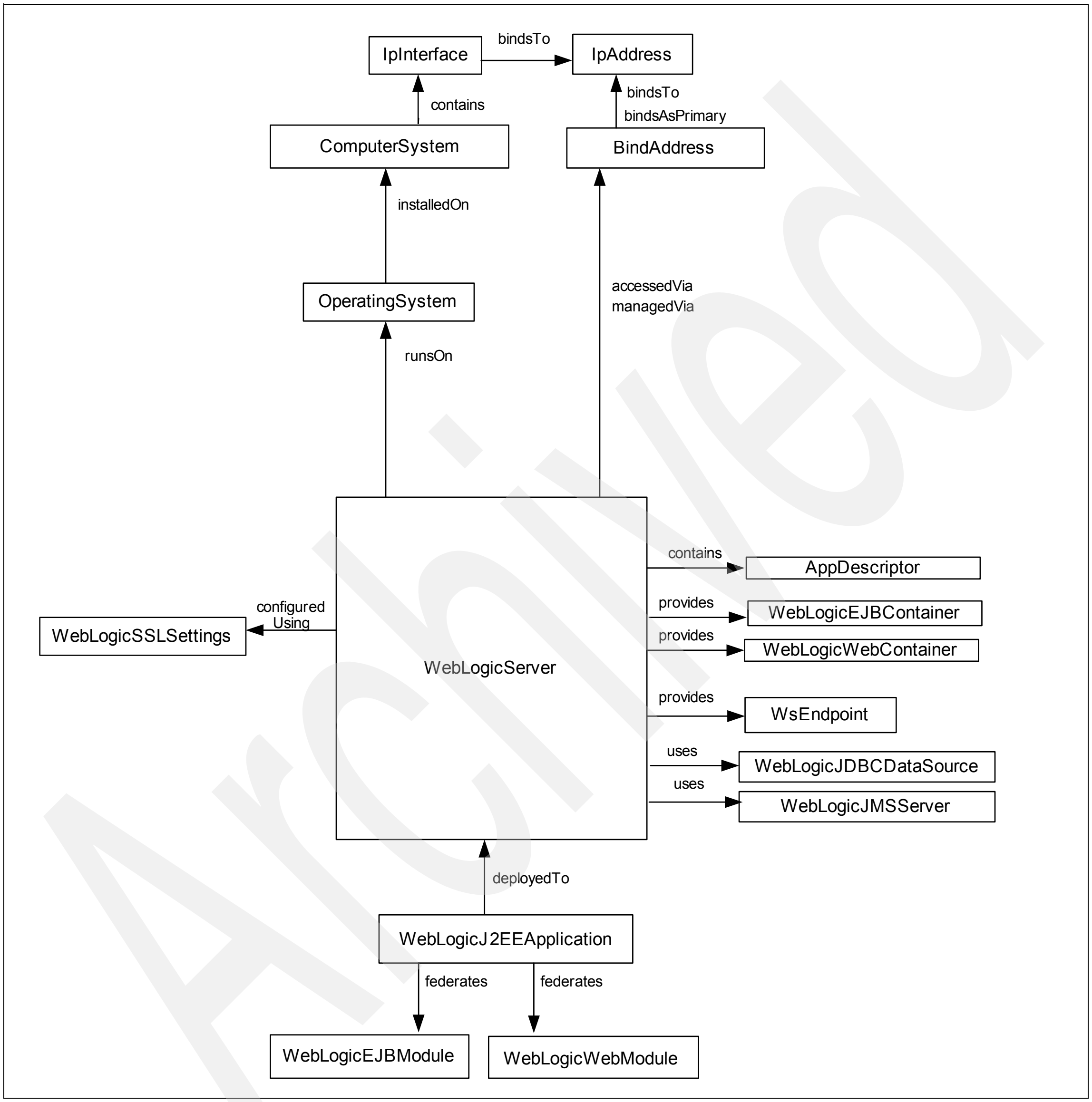

Figure 8   Instance diagram, WebLogic single-node server

Important relationships

Table 11 shows the important relationships in this scenario.

Table 11 Important relationships, WebLogic single-node server

Source class	Relationship type	Target class	Cardinality
BindAddress	bindsAsPrimary	IpAddress	1:1
BindAddress	bindsTo	IpAddress	1:1
ComputerSystem	contains	IpInterface	1:m
IpInterface	bindsTo	IpAddress	1:m
OperatingSystem	installedOn	ComputerSystem	m:1
WebLogicJ2EEApplication	deployedTo	WebLogicServer	m:n
WebLogicJ2EEApplication	federates	WebLogicEJBModule	m:n
WebLogicJ2EEApplication	federates	WebLogicWebModule	m:n
WebLogicServer	accessedVia	BindAddress	1:1
WebLogicServer	configuredUsing	WebLogicSSLSettings	1:1
WebLogicServer	contains	AppDescriptor	1:m
WebLogcServer	managedVia	BindAddress	1:1
WebLogicServer	provides	WebLogicEJBContainer	1:m
WebLogicServer	provides	WebLogicWebContainer	1:m
WebLogicServer	provides	WSEndpoint	1:m
WebLogicServer	runsOn	OperatingSystem	m:1
WebLogicServer	uses	WebLogicJDBCDataSource	1:m
WebLogicServer	uses	WebLogicJMSServer	1:m

Potential questions

Here are some related questions:

- On what IP address and port is the WebLogic server listening?
- What are the J2EE applications that are deployed to this server?
- What EJB modules and Web modules make up these applications?
- What operating system and computer system is the WebLogic server running on?
- What Web service endpoints are provided by this server, and what are the IP address and port for each Web Service endpoint?
- What are the JNDI addresses for the data sources and JMS servers used by the WebLogic server?
- What JDBC driver (implementation class name) is being used for each datasource?
- What databases are being accessed by each datasource?

WebLogic multi-node server

This scenario shows a WebLogic server that is a part of a cluster of servers. The server is a member of a domain, a domain federates a cluster, and a cluster federates the server. J2EE applications are deployed to either the server or to the cell. Otherwise, the description of the WebLogic single-node server in “WebLogic single-node server” on page 29 applies.

Naming rules and naming attributes

See for the naming rules and naming attributes of each class shown in Figure 9.

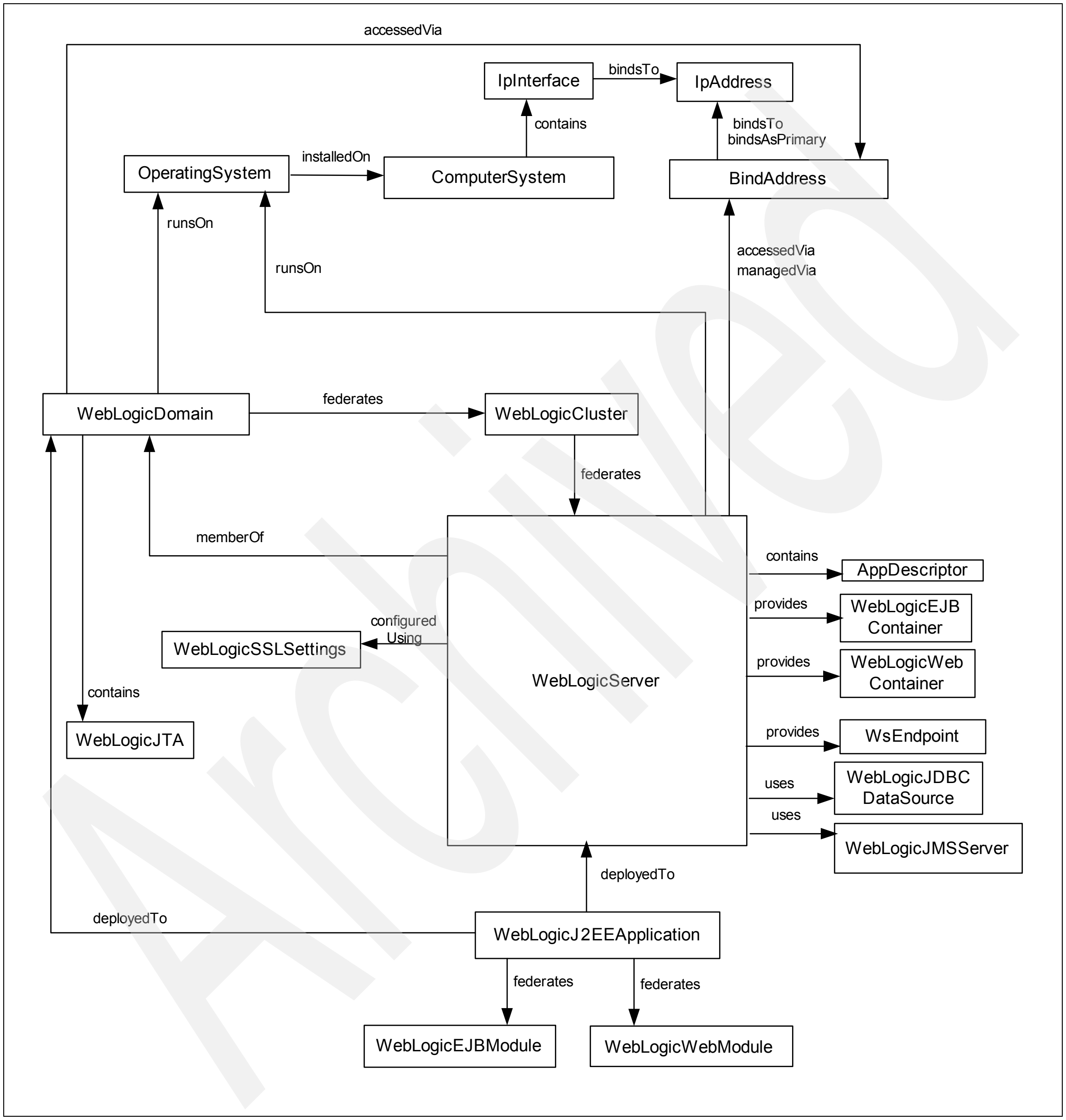

Figure 9 Instance diagram, WebLogic multi-node server

Important relationships

Table 12 shows the important relationships in this scenario. This list of important relationships is in addition to all of the relationships listed in “WebLogic single-node server” on page 29.

Table 12 Important relationships, WebLogic Multi-Node server

Source class	Relationship type	Target class	Cardinality
WebLogicCluster	federates	WebLogicServer	1:m
WebLogicDomain	accessedVia	BindAddress	1:1
WebLogicDomain	contains	WebLogicJTA	1:1
WebLogicDomain	federates	WebLogicCluste r	1:m
WebLogicJ2EEAppl ication	deployedTo	WebLogicDomai n	m:1
WebLogicMachine	contains	WebLogicNode Manager	1:1
WebLogicMachine	memberOf	WebLogicDomai n	m:1
WebLogicServer	memberOf	WebLogicDomai n	m:1
WebLogicServer	runsOn	WebLogicMachi ne	m:1

Potential questions

These queries are in addition to all of the queries listed in “WebLogic single-node server” on page 29.

- Which cluster and domain is the WebLogic server a member of?
- Which applications are deployed to the domain?
- What other WebLogic servers are parts of the same domain?
- What other WebLogic servers are parts of the same cluster?

Database server scenarios

This section presents CDM scenarios for four database servers: DB2, DB2 (zOS), Oracle, and Microsoft SQL Server™.

DB2

This scenario shows a simple DB2 system on a non-zOS system. The DB2 system contains a DB2 instance and an administrative server. The DB2 instance contains a database that contains schemas, buffer pools, and table spaces.

Naming rules and naming attributes

See for the naming rules and naming attributes of each class shown in Figure 10.

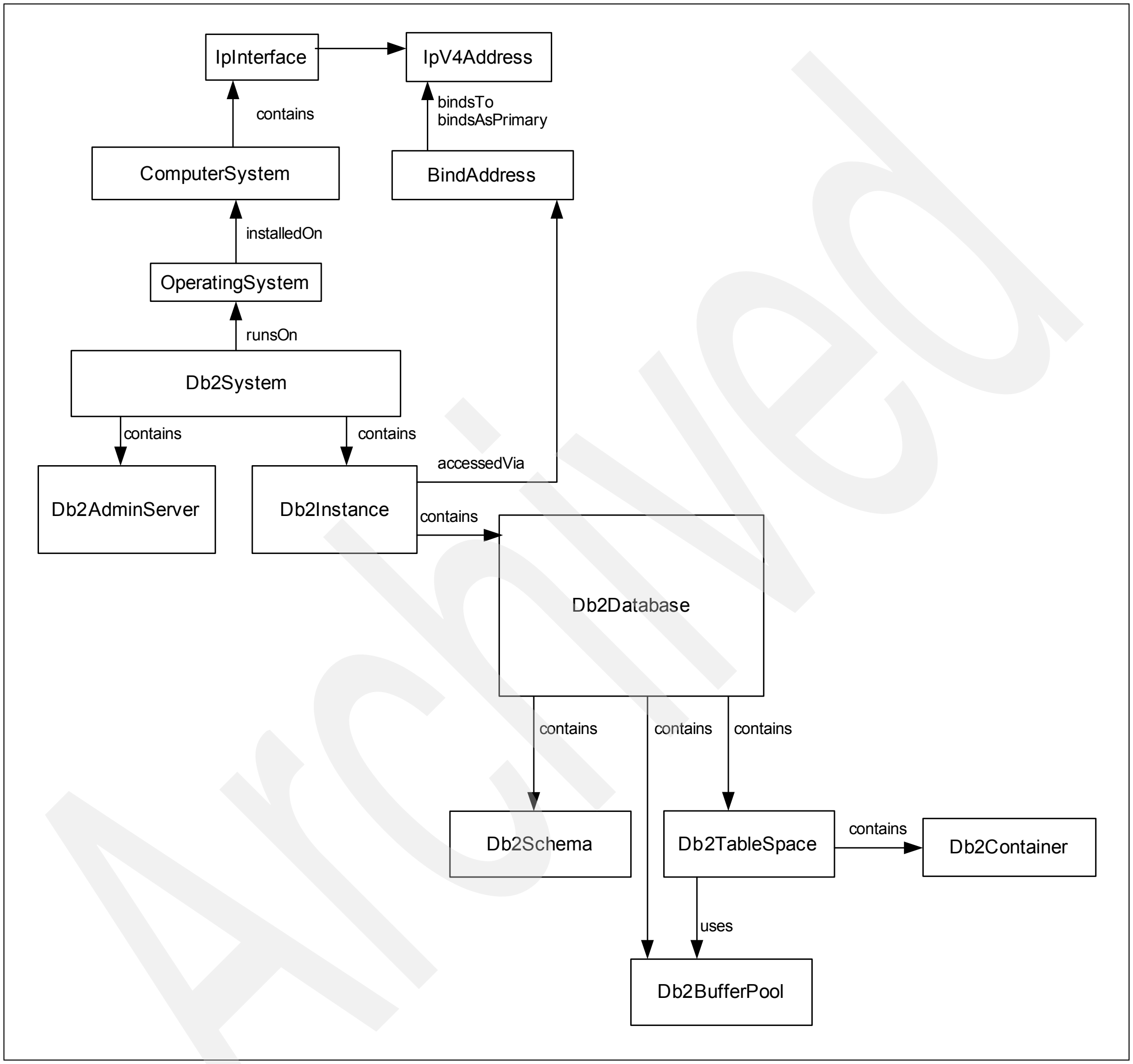

Figure 10   Instance diagram, DB2 database server

Important relationships

Table 13 shows the important relationships in this scenario.

Table 13 Important relationships, DB2 database server

Source class	Relationship type	Target class	Cardinality
BindAddress	bindsAsPrimary	IpV4Address	1:1
BindAddress	bindsTo	IpV4Address	1:1
ComputerSystem	contains	IpInterface	1:m
Db2Database	contains	Db2BufferPool	1:m
Db2Database	contains	Db2Schema	1:m
Db2Database	contains	Db2TableSpace	1:m
Db2Instance	accessedVia	BindAddress	1:1
Db2Instance	contains	Db2Database	1:m
Db2System	contains	Db2AdminServer	1:1
Db2System	contains	Db2Instance	1:m
Db2System	runsOn	OperatingSystem	m:1
Db2TableSpace	contains	Db2Container	1:m
Db2TableSpace	uses	Db2BufferPool	m:1
IpInterface	bindsTo	IpAddress	1:m
OperatingSystem	installedOn	ComputerSystem	m:1

Potential questions

Here are some related questions:

- On what IP address and port is the DB2 instance listening?
- On what operating system and computer system is the DB2 system running?
- What schemas are contained in this DB2 database?
- What other DB2 databases are contained in this DB2 instance?
- What other DB2 databases are contained in this DB2 system?
- Is JDBC tracing enabled for the DB2 database?

- What buffer pools and table spaces are used by the DB2 database?
- What is the page size for each buffer pool and tablespace?

DB2 (zOS)

This scenario shows a DB2 database that is running on zOS. The description in "DB2 " on page 35 applies. In addition, a DB2 sharing group contains the database and buffer pools, and federates DB2 subsystems. A DB2 subsystem contains the database and federates zOS address spaces.

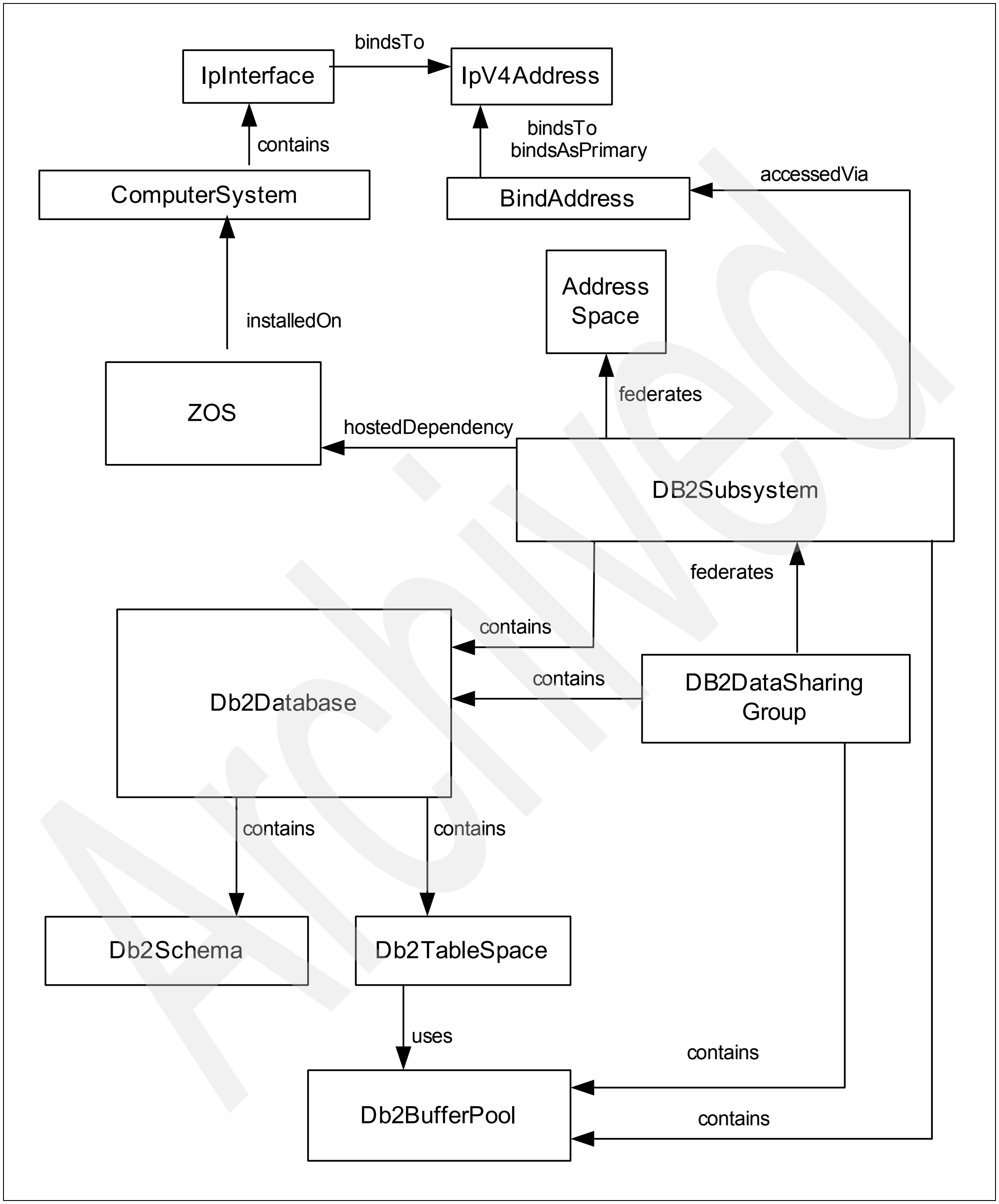

Figure 11 Instance diagram, DB2 (zOS) database server

Naming rules and naming attributes

See for the naming rules and naming attributes of each class shown in Figure 11 on page 39.

Important relationships

Table 14 shows the important relationships in this scenario.

Table 14 Important relationships, DB2 (zOS) database server

Source class	Relationship type	Target class	Cardinality
BindAddress	bindsAsPrimary	IpV4Address	1:1
BindAddress	bindsTo	IpV4Address	1:1
ComputerSystem	contains	IpInterface	1:m
Db2Database	contains	Db2Schema	1:m
Db2Database	contains	Db2TableSpace	1:m
DB2DataSharing Group	contains	Db2BufferPool	1:m
DB2DataSharing Group	contains	Db2Database	1:m
DB2DataSharing Group	federates	DB2Subsystem	1:m
DB2Subsystem	accessedVia	BindAddress	m:1
DB2Subsystem	contains	Db2BufferPool	1:1
DB2Subsystem	contains	Db2Database	1:m
DB2Subsystem	federates	AddressSpace	m:1
DB2Subsystem	hostedDependen cy	ZOS	m:1
Db2TableSpace	uses	Db2BufferPool	m:1
IpInterface	bindsTo	IpAddress	1:m
OperatingSystem	installedOn	ComputerSystem	m:1

Potential questions

These queries are in addition to all of the queries listed in “DB2 ” on page 35:

- Which data sharing group contains this database?
- Which DB2 subsystem contains this database?
- What is the version/release of this DB2 subsystem?
- What DB2 databases are contained in this DB2 subsystem and data sharing group?
- Which zOS hosts the DB2 subsystem?

Oracle

This scenario shows an Oracle database. The database contains schemas, data files, table spaces, and so on. It contains a database link to other Oracle databases. It also contains a control file and a redo log file. An instance contains the database, along with a server process and background processes. An Oracle server contains the instance. The system runs on an operating system.

Naming rules and naming attributes

See for the naming rules and naming attributes of each class shown in Figure 12.

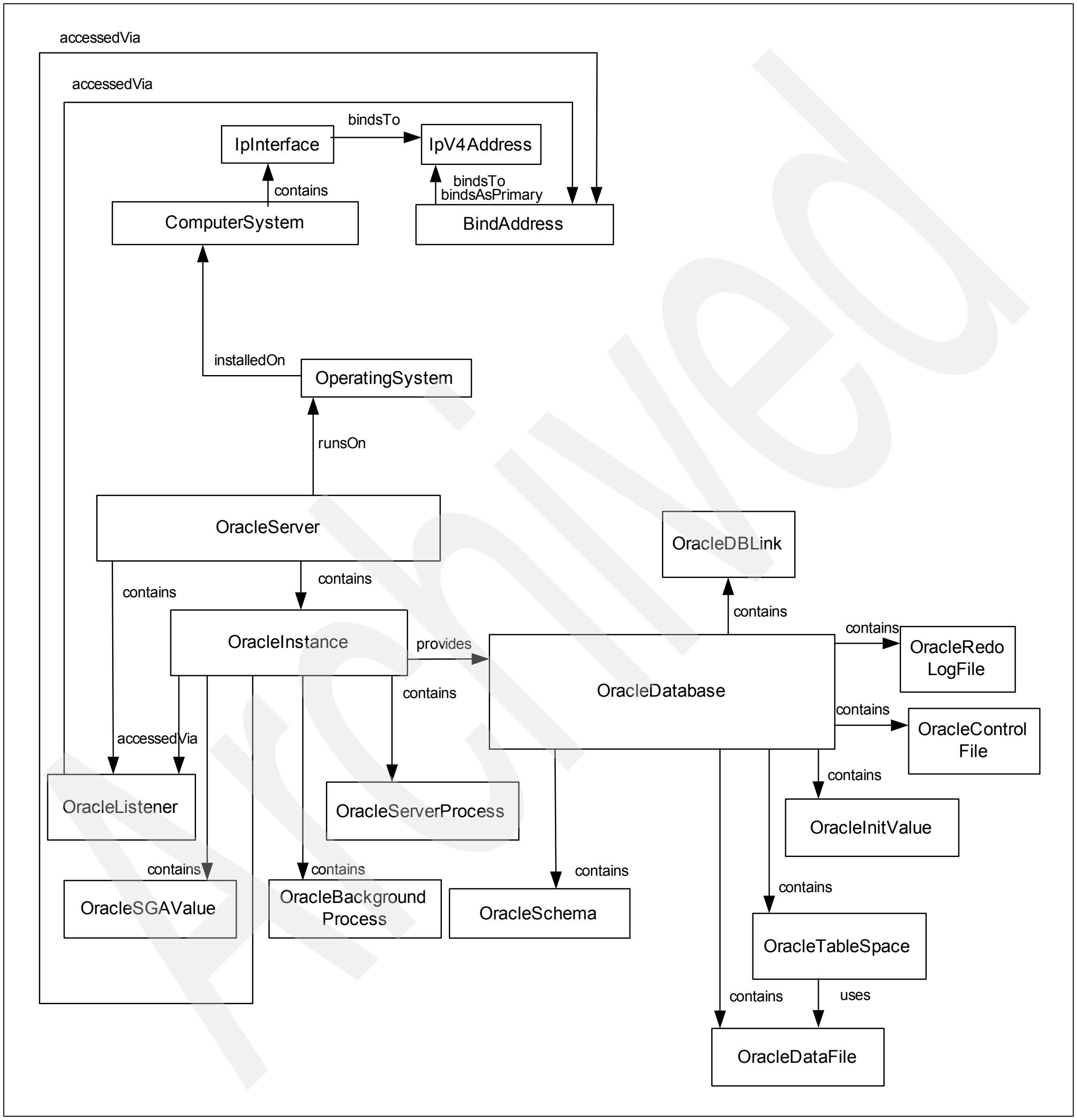

Figure 12   Instance diagram, Oracle database server

Important relationships

Table 15 shows the important relationships in this scenario.

Table 15 Important relationships, Oracle database server

Source class	Relationship type	Target class	Cardinality
BindAddress	bindsAsPrimary	IpV4Address	1:1
BindAddress	bindsTo	IpV4Address	1:1
ComputerSystem	contains	IpInterface	1:m
IpInterface	bindsTo	IpV4Address	1:m
OperatingSystem	installedOn	ComputerSystem	m:1
OracleDatabase	contains	OracleControlFile	1:m
OracleDatabase	contains	OracleDataFile	1:m
OracleDatabase	contains	OracleDBLink	1:m
OracleDatabase	contains	OracleInitValue	1:m
OracleDatabase	contains	OracleRedoLogFile	1:m
OracleDatabase	contains	OracleSchema	1:m
OracleDatabase	contains	OracleTableSpace	1:m
OracleInstance	accessedVia	BindAddress	1:1
OracleInstance	accessedVia	OracleListener	m:n
OracleInstance	contains	OracleBackgroundProcess	1:m
OracleInstance	contains	OracleServerProcess	1:m
OracleInstance	contains	OracleSGAValue	m:n
OracleInstance	provides	OracleDatabase	1:1
OracleListener	accessedVia	BindAddress	1:m
OracleServer	contains	OracleInstance	1:m
OracleServer	contains	OracleListener	1:m
OracleServer	runsOn	OperatingSystem	m:1
OracleTableSpace	uses	OracleDataFile	1:m

Potential questions

Here are some related questions:

- On what IP address and port is the Oracle instance listening?
- On what operating system and computer system is the Oracle server running?
- What schemas are contained by this Oracle database?
- What Oracle databases are contained in this Oracle instance?
- What Oracle databases are contained in this server?
- What level of tracing is enabled for the Oracle database?
- What table spaces are used by the Oracle database?
- What is the size of each tablespace?
- What is the size of the redo log file used by the Oracle database?
- What are the IP addresses and ports of the other Oracle databases that this Oracle database is linked to?

MS SQL

This scenario shows a Microsoft SQL Server database. The database contains tables. A SQL server contains the database, along with a server process. The server is configured using config values. The SQL server runs on an operating system.

Naming rules and naming attributes

See for the naming rules and naming attributes of each class shown in Figure 13.

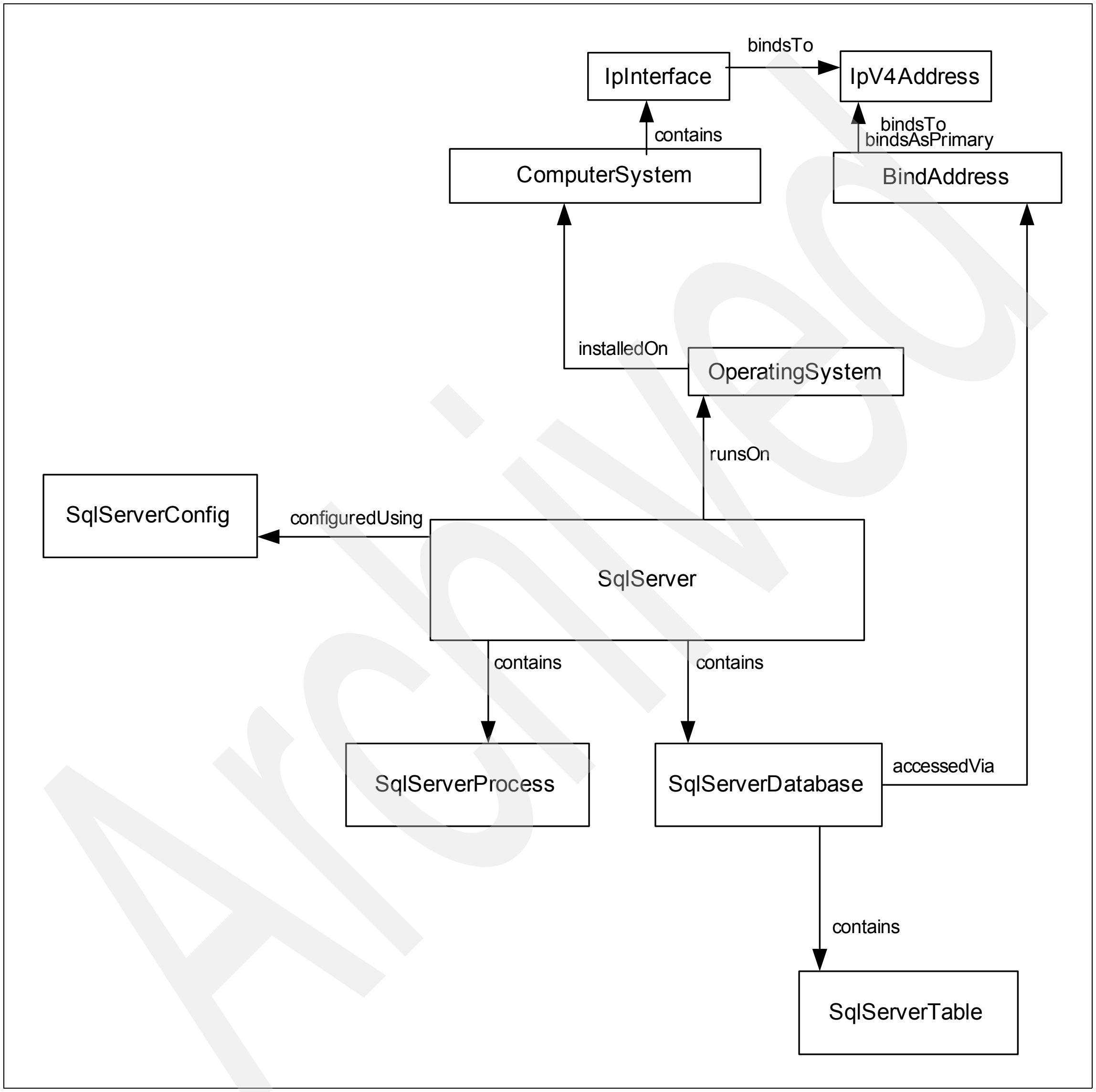

Figure 13   Instance diagram, MS SQL database server

Important relationships

Table 16 shows the important relationships in this scenario.

Table 16 Important relationships, Microsoft SQL Server database server

Source class	Relationship type	Target class	Cardinality
BindAddress	bindsAsPrimary	IpV4Address	1:1
BindAddress	bindsTo	IpV4Address	1:1
ComputerSystem	contains	IpInterface	1:m
IpInterface	bindsTo	IpV4Address	1:m
OperatingSystem	installedOn	ComputerSystem	m:1
SqlServer	configuredUsing	SqlServerConfig	1:m
SqlServer	contains	SqlServerDatabase	1:m
SqlServer	contains	SqlServerProcess	1:m
SqlServer	runsOn	OperatingSystem	m:1
SqlServerDatabase	accessedVia	BindAddress	1:1
SqlServerDatabase	contains	SqlServerTable	1:m

Potential questions

Here are some related questions:

- On which IP address and port is the SQL Server database listening?
- Which tables are contained by the database?
- When was the database created?
- What is the version of the database?
- Which server contains the database?
- On which operating system and computer system is the server running?

Network scenarios

This section presents CDM scenarios for three networks: a standard, single-room IT data center network topology; a standard building-site network topology; and a standard, global, multi-site wide-area-network trunk.

Standard, single-room IT data center network topology

This scenario shows a network with a single router that connects multiple computer systems. Connectivity is shown at both the IpInterface layer (layer 3) and at the L2Interface layer (layer 2). Note that this scenario will work without the L2Interface class if L2Interface data is not available. The classes that are directly associated with the router are shaded. For additional modeling details, see "Standard building-site network topology" on page 49, which includes a firewall. See "Server hardware" on page 72 for details on modeling the internal hardware of a router.

Naming rules and naming attributes

See for the naming rules and naming attributes of each class shown in Figure 14 on page 48.

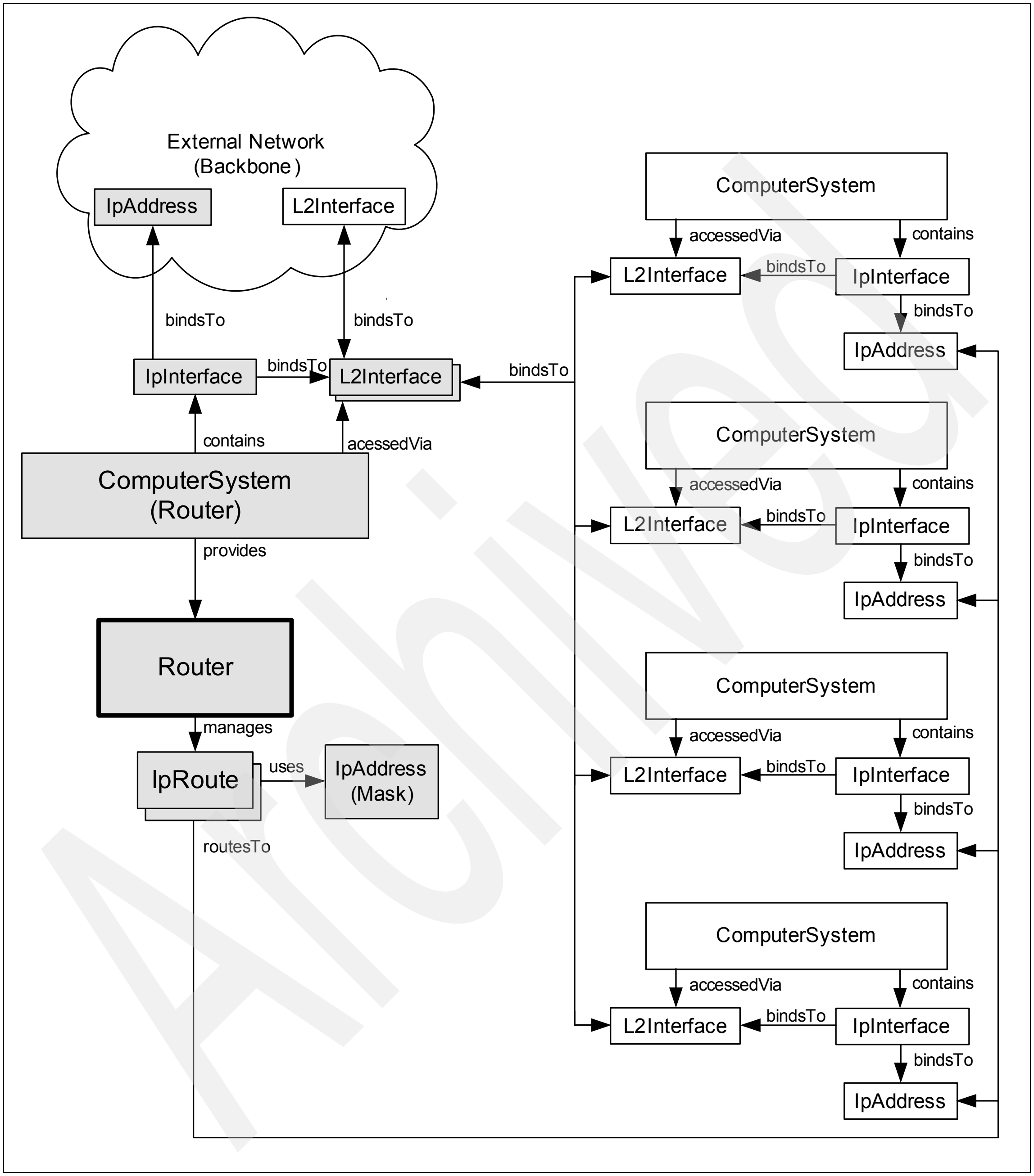

Figure 14   Instance diagram, standard, single-room IT data center network topology

Important relationships

Table 17 shows the important relationships in this scenario.

Table 17 Important relationships, standard, single-room IT data center network

Source class	Relationship type	Target class	Cardinality
ComputerSystem	accessedVia	L2Interface	1:m
ComputerSystem	contains	IpInterface	1:m
ComputerSystem	provides	Router	1:m
IpInterface	bindsTo	IpAddress	1:m
IpInterface	bindsTo	L2Interface	m:1
IpRoute	routesTo	IpAddress	1:1
IpRoute	uses	IpAddress	m:1
L2Interface	bindsTo	L2Interface	1:m
Router	manages	IpRoute	1:m

Usage or implementation notes

The Speed attribute of L2Interface can be used to determine bandwidth.

The HwAddress attribute of L2Interface can be used to determine the MAC address.

Potential questions

Here are some related questions:

- How are computers connected together in a given network?
- If a router goes down, which computers are no longer accessible?
- Is a particular computer system online?
- Which ports are open on a computer system?
- What is the bandwidth of a particular connection?
- What are the primary MAC addresses of the systems behind this router?
- What is the subnet mask?

Standard building-site network topology

This scenario shows a basic network topology for a building with a router/switch on each of three floors and an optional firewall on the external gateway. This scenario can be scaled as needed. Here the L2Interfaces would likely represent fiber interfaces. For clarity the layer 2 and layer 3 (IP) connectivity is only detailed between the gateway and the first floor switch. See “Standard, single-room IT data center network topology” on page 47 for details on the specific router/switch connectivity to the computers on each floor. The classes that are directly associated with the gateway are shaded.

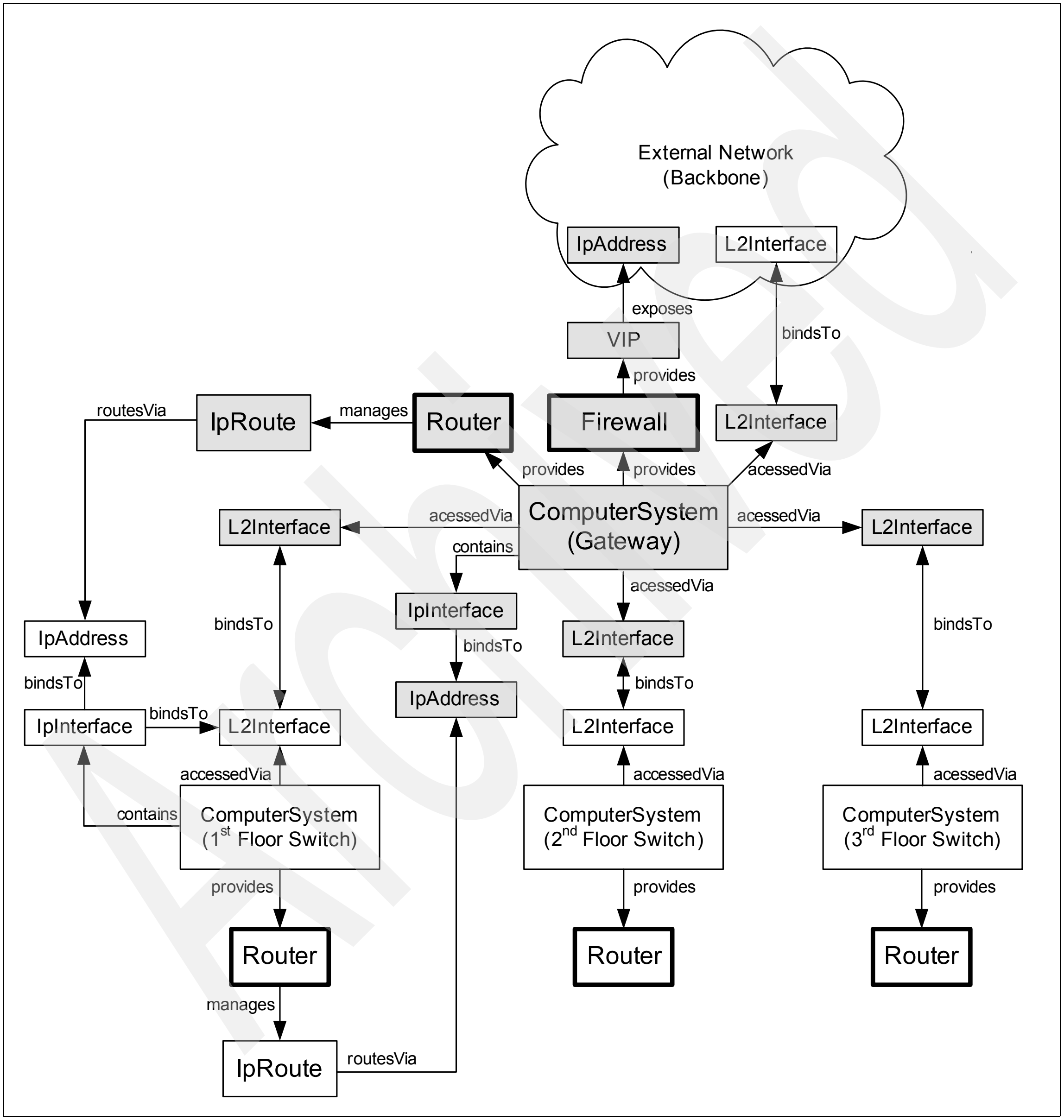

Figure 15   Instance diagram, standard building-site network topology

Naming rules and naming attributes

See for the naming rules and naming attributes of each class shown in Figure 15 on page 50.

Important relationships

Table 18 shows the important relationships in this scenario.

Table 18 Important relationships, standard building-site network topology

Source class	Relationship type	Target class	Cardinality
ComputerSystem	accessedVia	L2Interface	1:m
ComputerSystem	contains	IpInterface	1:m
ComputerSystem	provides	Firewall	1:m
ComputerSystem	provides	Router	1:m
Firewall	provides	VIP	1:m
IpInterface	bindsTo	IpAddress	1:m
IpInterface	bindsTo	L2Interface	m:1
IpRoute	routesVia	IpAddress	m:1
L2Interface	bindsTo	L2Interface	1:m
Router	manages	IpRoute	1:m
VIP	exposes	IpAddress	m:1

Usage or implementation notes

The Speed attribute of L2Interface can be used to determine bandwidth.

The HwAddress attribute of L2Interface can be used to determine the MAC address.

Potential questions

Here are some related questions:

- Which systems are behind a given firewall?
- How are computers connected together in a given network?
- If a router/firewall goes down, which computers are no longer accessible?
- What is the bandwidth of a particular connection?
- What are the primary MAC addresses of the systems behind a router/switch?
- What is the subnet mask?

- What is the firewall's external IP address and its subnet mask?
- What is the internal subnet mask used by the firewall?

Standard, global, multi-site wide-area-network trunk

This scenario shows four geographically dispersed gateways. For specific details of the gateway configurations and the internal networks, see "Standard, single-room IT data center network topology" on page 47.

Naming rules and naming attributes

See "Naming rules and naming attributes " on page 78 for the naming rules and naming attributes of each class shown in Figure 16.

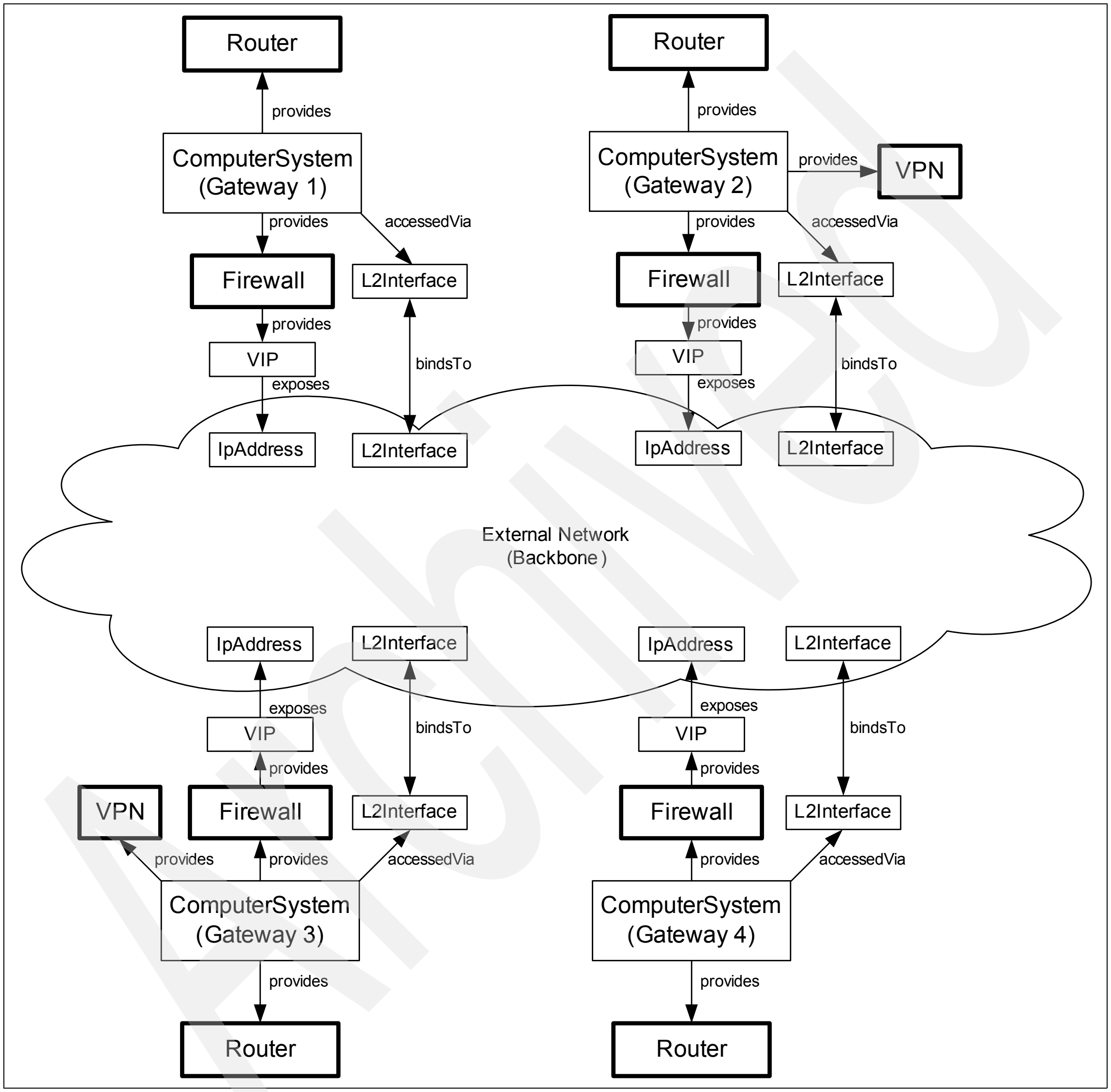

Figure 16 Instance diagram, standard, global, multi-site wide-area-network trunk

Important relationships

Table 19 shows the important relationships in this scenario.

Table 19 Important relationships, standard, global, multi-site wide-area-network trunk

Source class	Relationship type	Target class	Cardinality
ComputerSystem	accessedVia	L2Interface	1:m
ComputerSystem	provides	Firewall	1:m
ComputerSystem	provides	Router	1:m
ComputerSystem	provides	VPN	1:m
Firewall	provides	VIP	1:m
L2Interface	bindsTo	L2Interface	1:m
VIP	exposes	IpAddress	m:1

Usage or implementation notes

The Speed attribute of L2Interface can be used to determine bandwidth.

The HwAddress attribute of L2Interface can be used to determine the MAC address.

Potential questions

Some potential questions are:

- Which gateways are connected to the WAN backbone?
- What is the bandwidth of each gateway?

Storage scenarios

This section presents CDM scenarios for three storage area networks (SANs): a SAN with fabric topology and a zoned SAN with fabric topology.

Storage area network (fabric topology)

This scenario shows a fabric Fibre Channel SAN with two servers and two disk systems. These two servers share the two disk systems.

Naming rules and naming attributes

See “Naming rules and naming attributes ” on page 78 for the naming rules and naming attributes of each class shown in Figure 17.

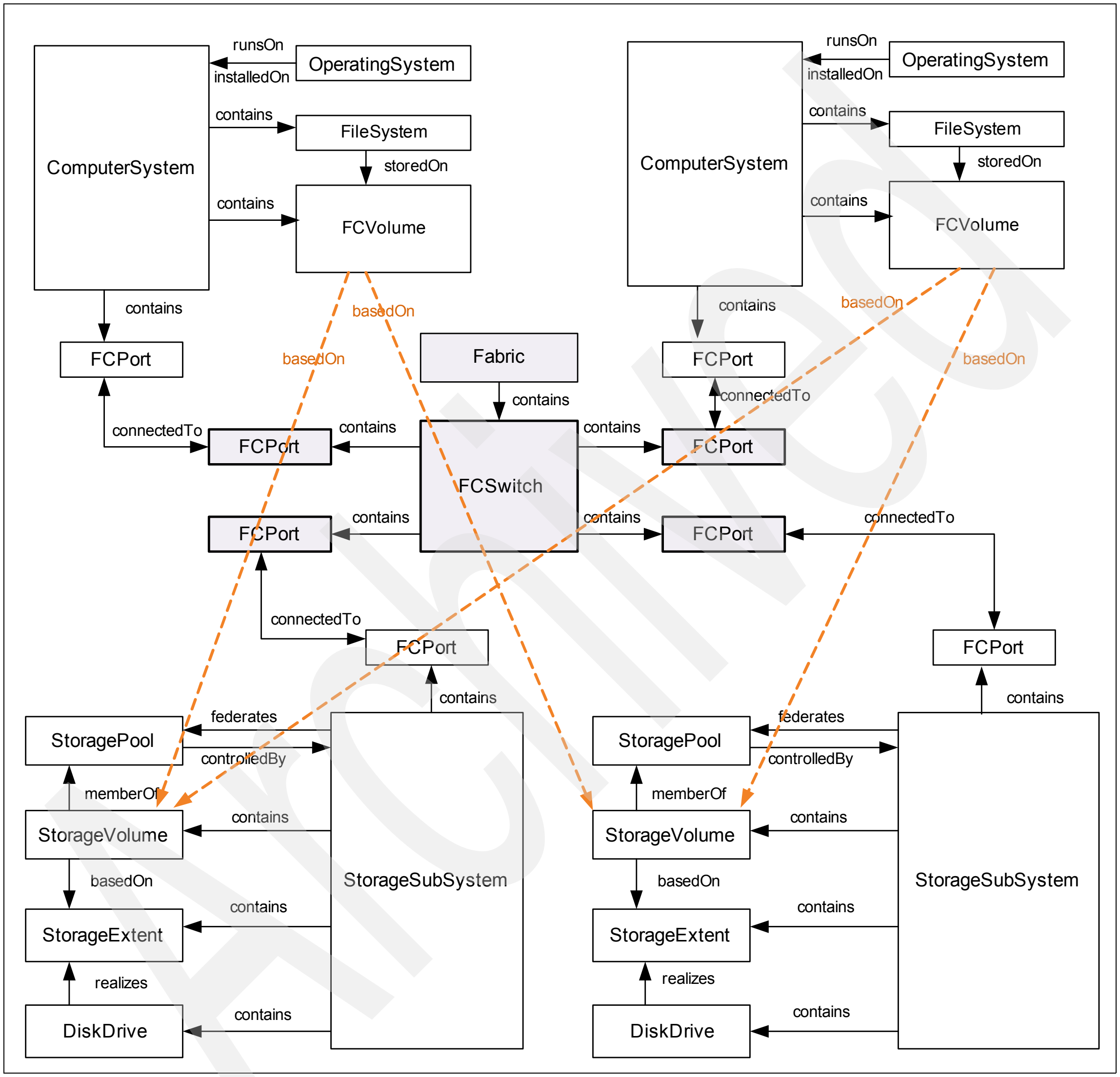

Figure 17 Instance diagram, SAN (Fabric topology)

Important relationships

Table 20 shows the important relationships in this scenario.

Table 20 Important relationships, storage area network with Fabric topology

Source class	Relationship type	Target class	Cardinality
ComputerSystem	contains	Filesystem	1:m
ComputerSystem	contains	FCVolume	1:m
ComputerSystem	contains	FCPort	1:m
DiskDrive	realizes	StorageExtent	m:n
Fabric	contains	FCSwitch	1:m
FCPort	connectedTo	FCPort	m:n
FCSwitch	contains	FCPort	1:m
FCVolume	basedOn	StorageVolume	m:n
FileSystem	storedOn	FCVolume	m:1
OperatingSystem	runsOn	ComputerSystem	1:1
OperatingSystem	installedOn	ComputerSystem	m:1
StoragePool	controlledBy	StorageSubsystem	m:1
StorageSubsystem	contains	FCPort	1:m
StorageSubsystem	contains	StorageExtent	1:m
StorageSubsystem	contains	DiskDrive	1:m
StorageSubsystem	federates	StoragePool	m:n
StorageSubsystem	contains	StorageVolume	1:m
StorageVolume	memberOf	StoragePool	m:n
StorageVolume	basedOn	StorageExtent	m:1

Potential questions

Here are some related questions:

- What servers use this SAN to share storage resources?
- What disk systems are connected to this SAN?
- What is the total disk space available in this SAN?
- How much free disk space is left in this SAN?
- How much free disk space is on this disk system?
- If this disk system is removed, which server will be impacted?
- What servers are connected to this storage subsystem?
- What servers are connected to this Fibre Channel switch?

Zoned storage area network (fabric topology)

This scenario shows a SAN Fabric network that contains four servers and four disk systems. This SAN is divided into two zones, and each zone contains two disk systems. The first server is a member of both zones and thus can access all four disk systems, while the other servers are members of either the first or the second zone, and thus can only access the disk systems in the same zone.

For clarity, in the instance diagram of this scenario, only the ComputerSystem class is shown for a server and only the StorageSubSystem class is shown for a disk system. See "Operating system server scenarios" on page 10 for details of a server, and "Storage area network (fabric topology)" on page 54 for details of a disk system.

Naming rules and naming attributes

See “Naming rules and naming attributes ” on page 78 for the naming rules and naming attributes of each class shown in Figure 18.

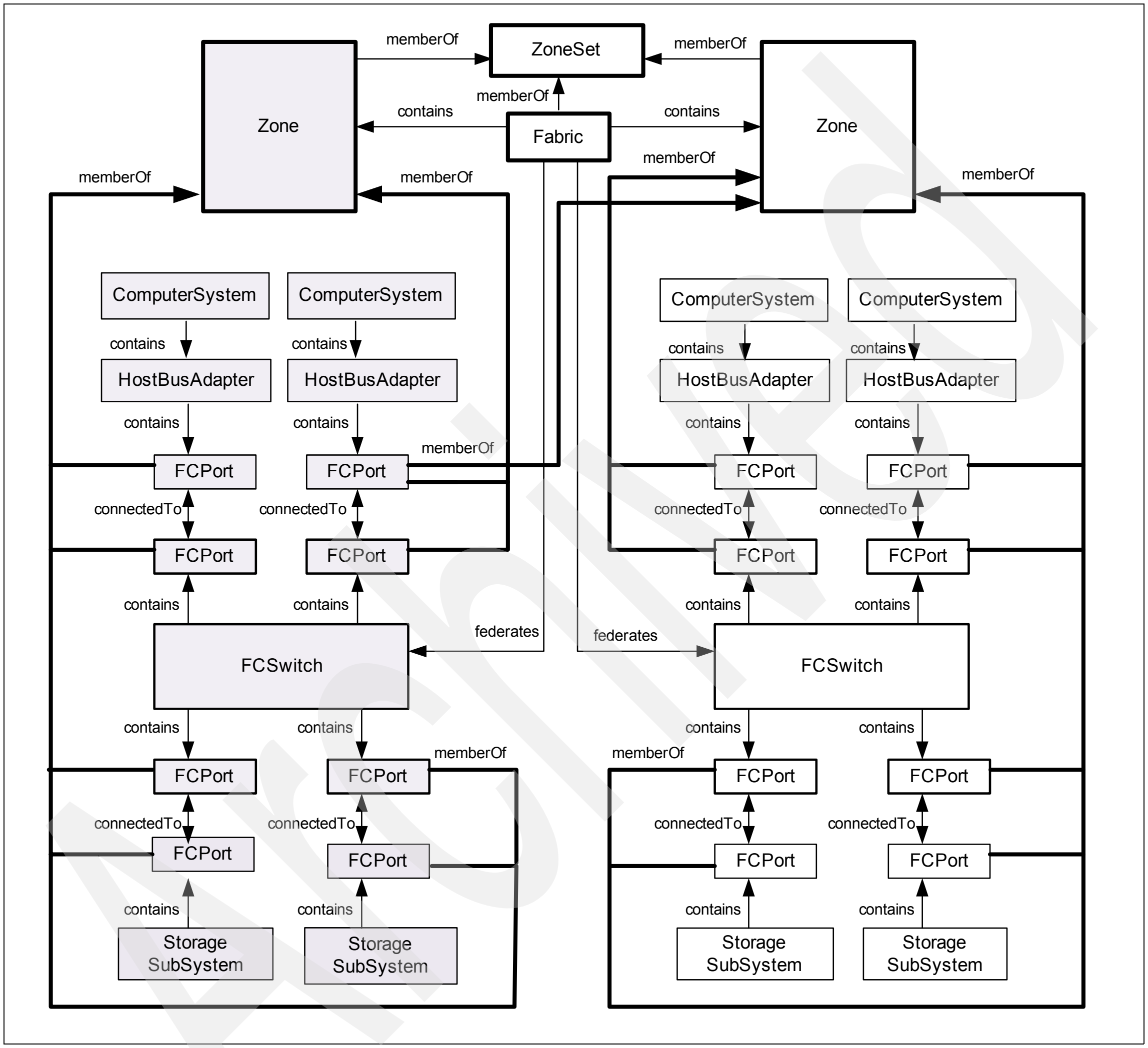

Figure 18 Instance diagram, zoned SAN (fabric topology)

Important relationships

Table 21 shows the important relationships in this scenario.

Table 21 Important relationships, zoned storage area network (fabric topology)

Source class	Relationship type	Target class	Cardinality
ComputerSystem	contains	HostBusAdapter	1:m
Fabric	contains	Zone	1:m
Fabric	memberOf	ZoneSet	m:n
Fabric	federates	FCSwitch	1:m
FCPort	connectedTo	FCPort	m:n
FCPort	memberOf	Zone	m:n
FCSwitch	contains	FCPort	1:m
HostBusAdapter	contains	FCPort	1:m
StorageSubsystem	contains	FCPort	1:m
Zone	memberOf	ZoneSet	m:n

Potential questions

Some potential questions are:

- What servers use this SAN to share storage resources?
- What disk systems are connected to this SAN?
- What is the total disk space available in this SAN?
- How much free disk space left in this SAN?
- How much free disk space is on this disk system?
- If this disk system in zone 1 is removed, which server is impacted?
- How many zones are in this SAN? Which servers and disk systems are members of this zone?
- How much disk space is available to a server in this zone?
- Which servers can access all the zones in this SAN?
- How much free disk space is left in this zone?
- What storage subsystems and servers are parts of this zone?

- How many CPUs are available on a particular blade enclosure? What are the associated clock speeds?

Business system scenarios

This section presents CDM scenarios for three business systems: a single-process and single-server business system, a multi-process business system, and a business system involving Web services.

Single-process and single-server business system

This scenario shows a simple business system that contains only one business process and involves only one server. This business process contains four activities. Among these activities, the first one can occur in parallel with the other ones, while three of them must occur in sequence, as shown in Figure 19. Note that some activities use ManagedElement, which could be any CDM resources, for example, a DB2 server, a WAS server, a MQ subsystem, a database, a J2EE bean, a Web service, and so on.

Naming rules and naming attributes

See “Naming rules and naming attributes ” on page 78 for the naming rules and naming attributes of each class shown in Figure 19.

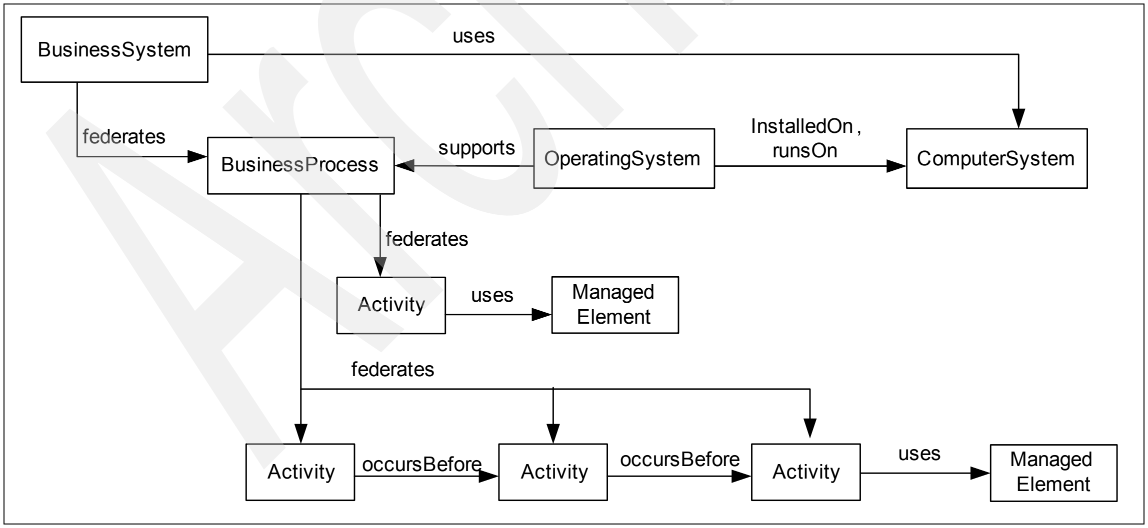

Figure 19 Instance diagram, single-process and single-server business system

Important relationships

Table 22 shows the important relationships in this scenario.

Table 22 Important relationships, single-process and single-server business system

Source class	Relationship type	Target class	Cardinality
Activity	occursBefore	Activity	m:n
Activity	uses	ManagedElement	m:n
BusinessProcess	federates	Activity	m:n
BusinessSystem	federates	BusinessProcess	m:n
BusinessSystem	uses	ComputerSystem	m:n
OperatingSystem	installedOn	ComputerSystem	m:1
OperatingSystem	runsOn	ComputerSystem	1:1
OperatingSystem	supports	BusinessProcess	m:n

Potential questions

Here are some related questions:

- When a server is down, what business system is impacted?
- Can I start this activity? What activities must be completed before this activity can start?

Multi-process business system

This scenario shows a business system that contains multiple business processes, and each business process involves a different server. Some activities of these business processes involve another business process. For example, the third business process of this business system contains three activities. The first and the third activities involve another business process.

Naming rules and naming attributes

See “Naming rules and naming attributes ” on page 78 for the naming rules and naming attributes of each class shown in Figure 20.

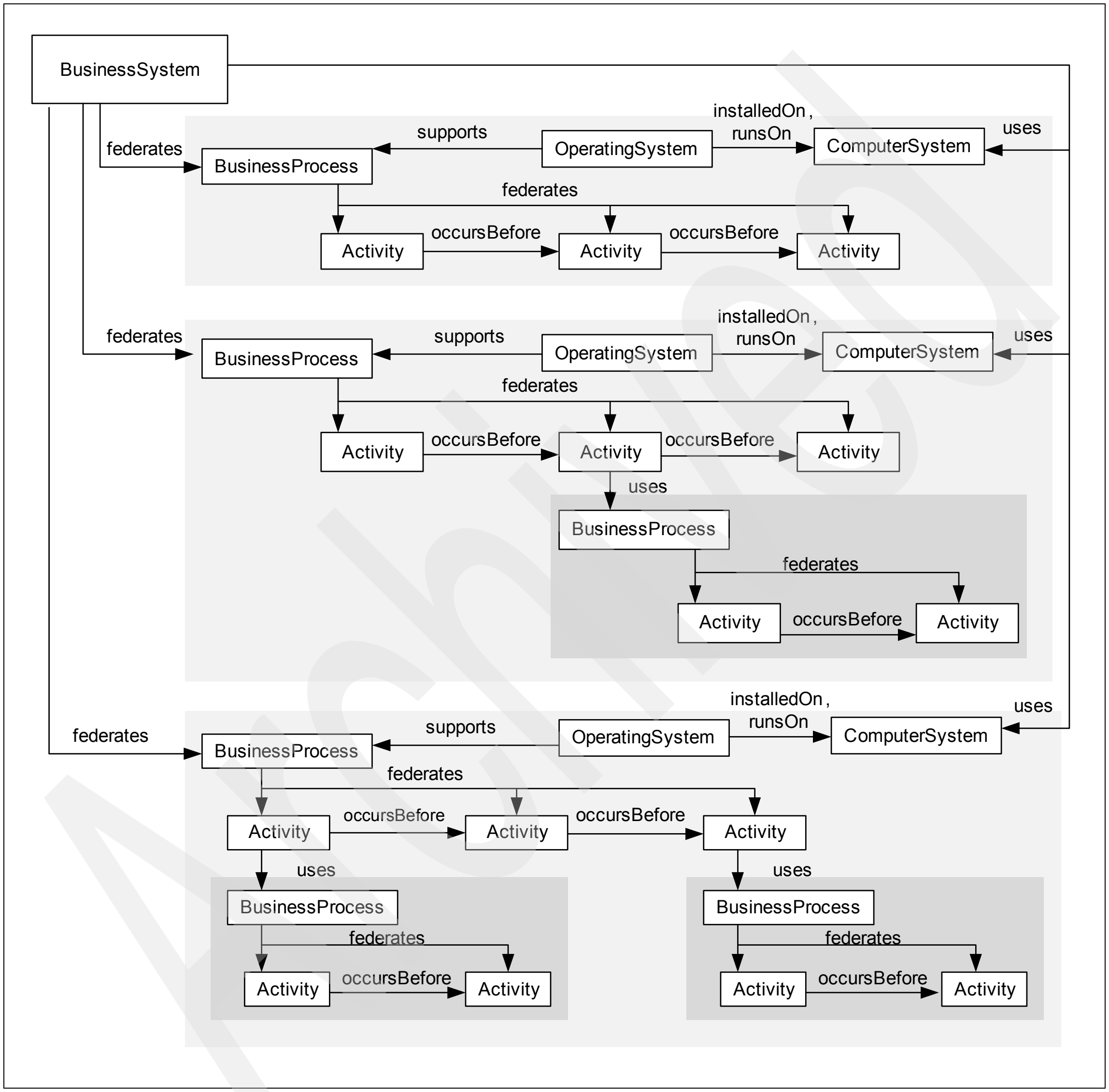

Figure 20 Instance diagram, multi-process business system

Important relationships

Table 23 shows the important relationships in this scenario.

Table 23 Important relationships, multi-process business system

Source class	Relationship type	Target class	Cardinality
Activity	occursBefore	Activity	m:n
Activity	uses	BusinessProcess	m:n
BusinessProcess	federates	Activity	m:n
BusinessSystem	federates	BusinessProcess	m:n
BusinessSystem	uses	ComputerSystem	m:n
OperatingSystem	installedOn	ComputerSystem	m:1
OperatingSystem	runsOn	ComputerSystem	1:1
OperatingSystem	supports	BusinessProcess	m:n

Potential questions

Here are some related questions:

- When a server is down, what business systems are impacted?
- Can I start this activity? What activities must be completed before this activity can start?

Business system involving Web services

This scenario shows a business system that contains multiple business processes. Some activities of these business processes invoke a Web service that resides on a different server. For details of a Web service, reference “Standard Web Services (WSDL) transaction” on page 70.

Naming rules and naming attributes

See "Naming rules and naming attributes " on page 78 for the naming rules and naming attributes of each class shown in Figure 21.

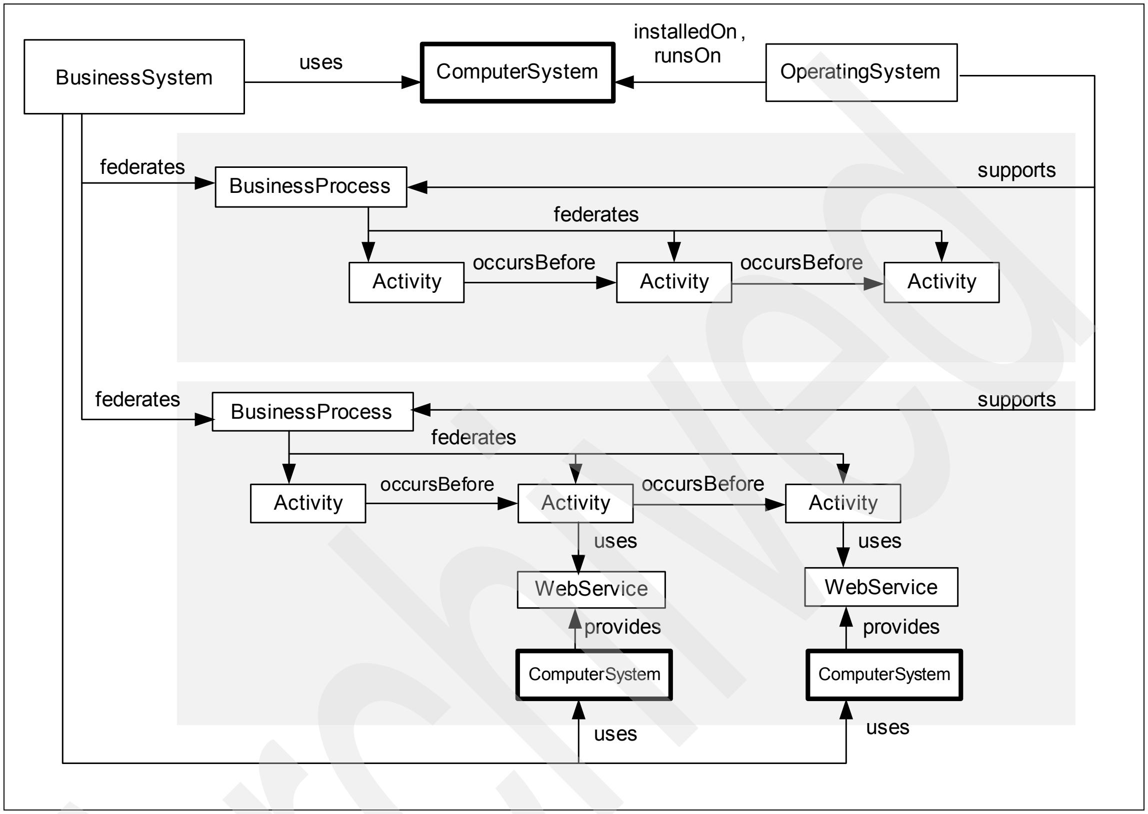

Figure 21 Instance diagram, business system involving Web services

Important relationships

Table 24 shows the important relationships in this scenario.

Table 24 Important relationships, business system involving Web services

Source class	Relationship type	Target class	Cardinality
Activity	occursBefore	Activity	m:n
Activity	uses	WebService	m:n
BusinessProcess	federates	Activity	m:n
BusinessSystem	federates	BusinessProcess	m:n

Source class	Relationship type	Target class	Cardinality
BusinessSystem	uses	ComputerSystem	m:n
ComputerSystem	provides	WebService	m:n
OperatingSystem	installedOn	ComputerSystem	m:1
OperatingSystem	runsOn	ComputerSystem	1:1
OperatingSystem	supports	BusinessProcess	m:n

Potential questions

Here are some related questions:

- When a server is down, what business systems are impacted?
- Can I start this activity? What activities must be completed before this activity can start?

Transaction scenarios

In CDM, a transaction is modeled via the Activity class. An instance of Activity represents a single *sub-transaction* in a larger overall BusinessProcess (or transaction, for example, a buy book transaction). There will be multiple instances of Activity in the larger overall transaction (for example, place order, check credit, and so on), each joined to its predecessor by the occursBefore CDM relationship. The transaction is not the EJB (or WebServer). Instead, it is the work that the EJB/WebServer is performing, represented as an Activity that *uses* the EBJ or the WebServer.

Simple transaction

This scenario shows a simple transaction that involves a Windows client, an HpUx server, and a zOS server. The entire transaction is modeled as a BusinessProcess that contains a sequence of five activities (or sub-transactions).

Naming rules and naming attributes

See “Naming rules and naming attributes ” on page 78 for the naming rules and naming attributes of each class shown in Figure 22.

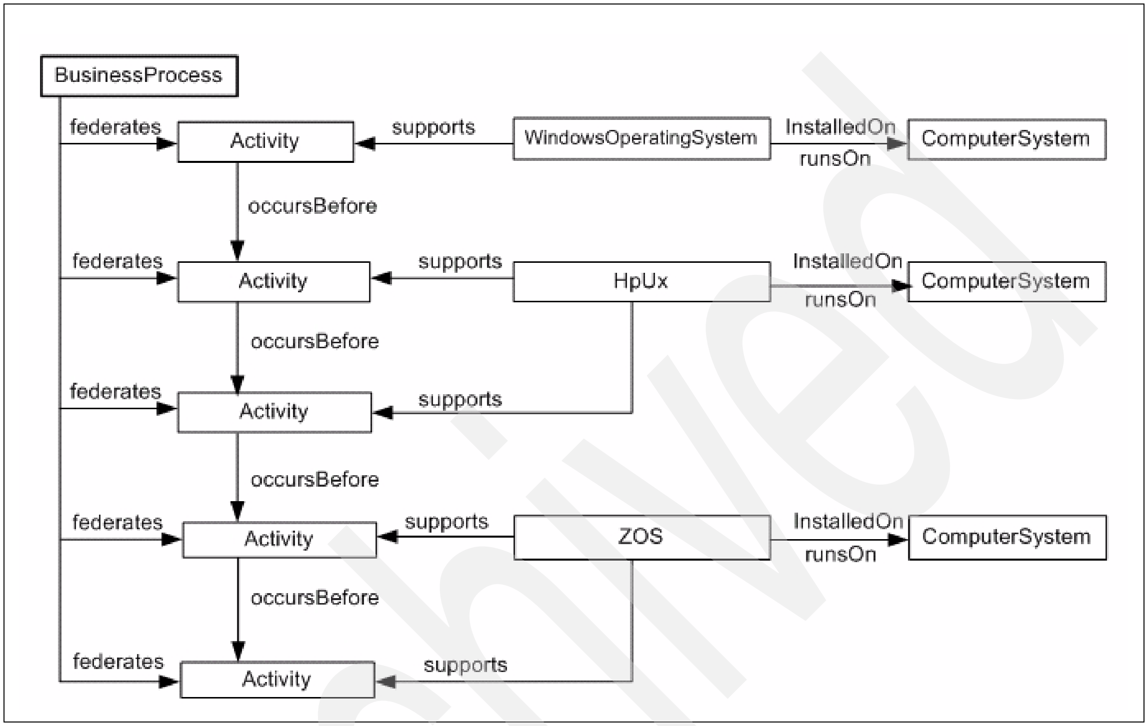

Figure 22 Instance diagram, simple transaction

Important relationships

Figure 23 on page 68 shows the important relationships in this scenario.

Table 25 Important relationships, simple transaction

Source class	Relationship type	Target class	Cardinality
Activity	occursBefore	Activity	m:n
BusinessProcess	federates	Activity	m:n
HpUx	installedOn	ComputerSystem	m:1
HpUx	runsOn	ComputerSystem	1:1
HpUx	supports	Activity	m:n

Source class	Relationship type	Target class	Cardinality
WindowsOperatingSystem	installedOn	ComputerSystem	m:1
WindowsOperatingSystem	runsOn	ComputerSystem	1:1
WindowsOperatingSystem	supports	Activity	m:n
ZOS	installedOn	ComputerSystem	m:1
ZOS	runsOn	ComputerSystem	1:1
ZOS	supports	Activity	m:n

Potential questions

Here are some related questions:

- What operating systems does this transaction depend on?
- If a server is down, what transactions are impacted?

Transaction (with resources)

This scenario shows a group of two transactions. The first transaction contains a sequence of three activities (or sub-transactions). These activities use the following resources: WebSphere Application server and DB2 server. The second transaction involves a sequence of two activities (or sub-transactions). One of the activities uses WebService. The resources (for example, WebSphere Application Server, DB2 server, Web Service, and so on) shown in Figure 23 on page 68 are examples. An activity can use other resources, such as IBM HTTP Server, IIS Server, J2EE domain, J2EE server, Web Logical Server, and so on.

Naming rules and naming attributes

See "Naming rules and naming attributes " on page 78 for the naming rules and naming attributes of each class shown in Figure 23.

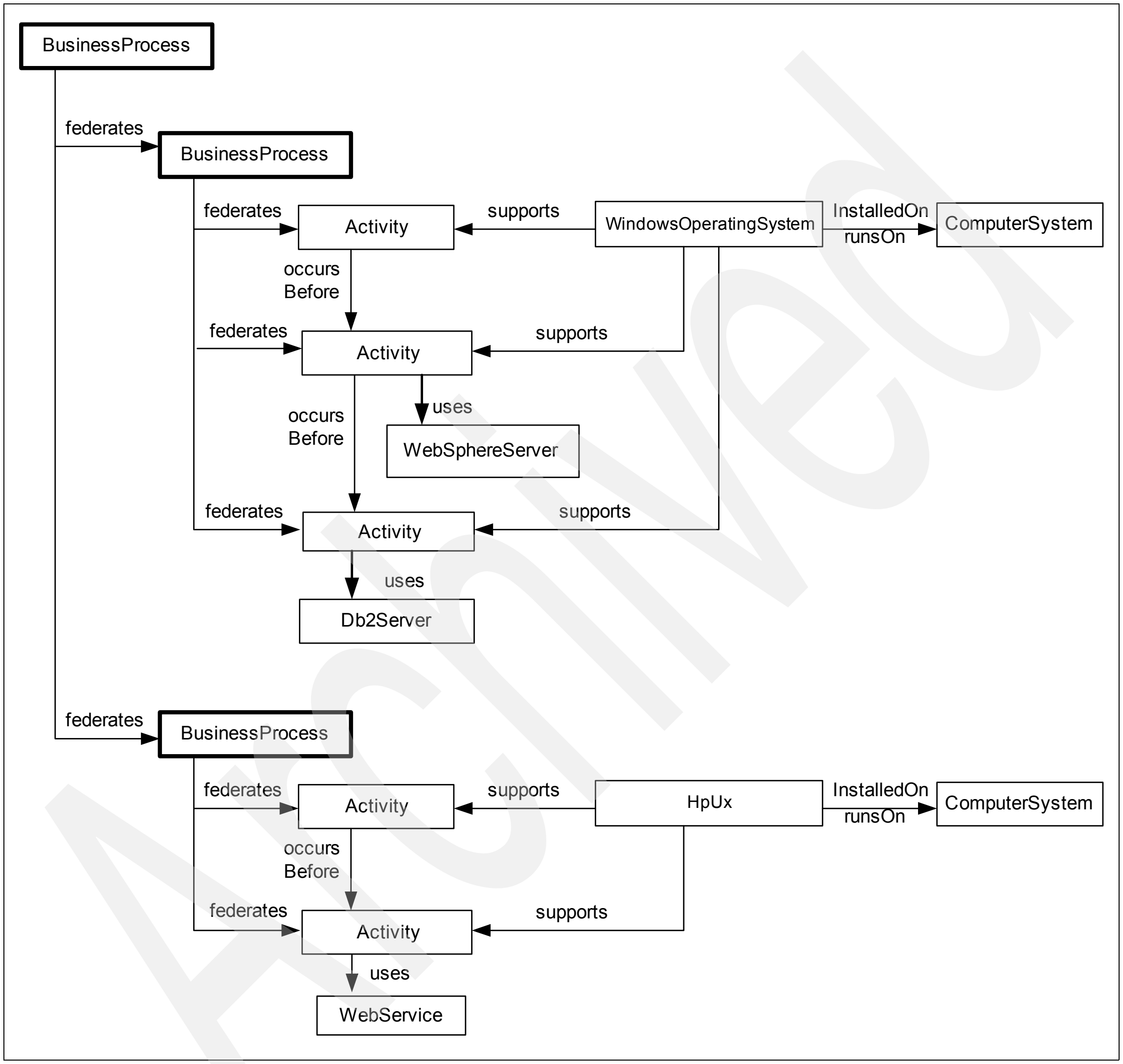

Figure 23 Instance diagram, transaction (with resources)

Important relationships

Table 26 shows the important relationships in this scenario.

Table 26 Important relationships, transaction (with resources)

Source class	Relationship type	Target class	Cardinality
Activity	occursBefore	Activity	m:n
Activity	uses	Db2Server	m:n
Activity	uses	WebSphereServer	m:n
Activity	uses	WebService	m:n
BusinessProcess	federates	Activity	m:n
BusinessProcess	federates	BusinessProcess	m:n
HpUx	installedOn	ComputerSystem	m:1
HpUx	runsOn	ComputerSystem	1:1
HpUx	supports	Activity	m:n
WindowsOperatingSystem	installedOn	ComputerSystem	m:1
WindowsOperatingSystem	runsOn	ComputerSystem	1:1
WindowsOperatingSystem	supports	Activity	m:n

Potential questions

Here are some related questions:

- What resources (for example, client, Web server, servlet, database server, or database) are involved in this transaction?
- What other transactions are involved in this transaction?
- What operating systems does this transaction depend on?
- If a server is down, what transactions are impacted?

Standard Web Services (WSDL) transaction

This scenario shows a typical Web service transaction. The IP V4 address shown in the diagram is an example. The IP V4 address shown in Figure 24 is an example. IP V6 is supported by replacing the use of the IpV4Address class with the IpV6Address class.

Naming rules and naming attributes

See "Naming rules and naming attributes " on page 78 for the naming rules and naming attributes of each class shown in Figure 24.

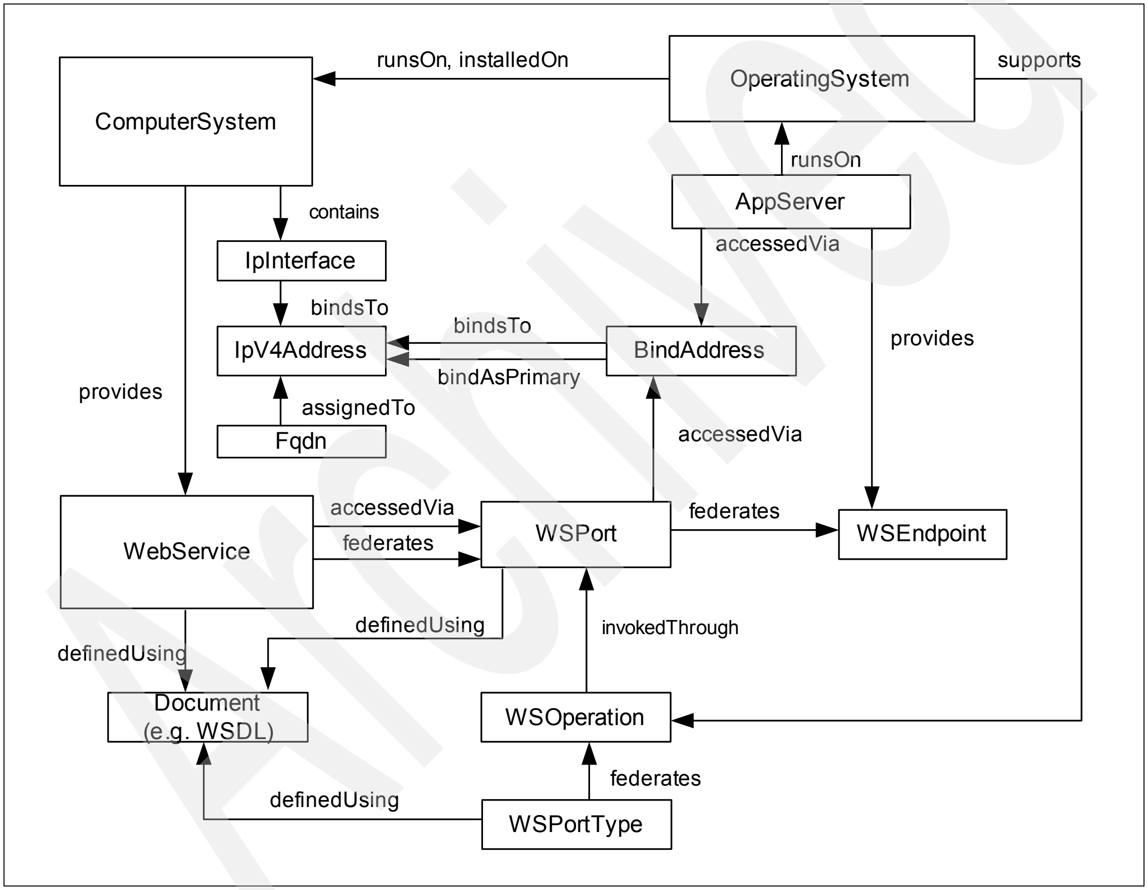

Figure 24 Instance diagram, standard Web services (WSDL) transaction

Important relationships

Table 27 shows the important relationships in this scenario.

Table 27 Important relationships, standard Web services (WSDL) transaction

Source class	Relationship type	Target class	Cardinality
AppServer	accessdVia	BindAddress	1:1
AppServer	runsOn	OperatingSystem	m:1
AppServer	provides	WSEndpoint	1:m
BindAddress	bindAsPrimary	IpV4Address	1:1
BindAddress	bindsTo	IpV4Address	1:1
ComputerSystem	contains	IpInterface	1:m
ComputerSystem	provides	WebService	m:n
Fqdn	assignedTo	IpV4Address	1:1
IpInterface	bindsTo	IpV4Address	1:m
OperatingSystem	installedOn	ComputerSystem	m:1
OperatingSystem	runsOn	ComputerSystem	1:1
OperatingSystem	supports	WSOperation	m:n
WebService	accessedVia	WSPort	1:m
WebService	definedUsing	Document	m:n
WebService	federates	WSPort	m:n
WSPort	accessedVia	BindAddress	m:1
WSPort	federates	WSEndpoint	1:m
WSPort	definedUsing	Document	m:n
WSPortType	definedUsing	Document	m:n
WSPortType	federates	WSOperation	m:n
WSOperation	invokedThrough	WSPort	m:n

Potential questions

Here are some related questions:

- What Web service is supported on this computer?
- What Web service operations are supported via this Web service port type?
- On what application servers was this Web service deployed?

Hardware scenarios

This section presents CDM scenarios for two hardware classes: server hardware and blade enclosure.

Server hardware

This scenario shows the physical/hardware aspects of a server. This shows a basic server configuration that is adaptable to multiple devices, multiple CPUs, disk drives, and so on. This model could be used to represent any server, including Windows, UNIX, and zSeries systems. The specific operating system class would be determined by the actual operating system installation. The model for a blade in the blade enclosure scenario (see “Blade enclosure” on page 75) could be used for an even simpler model of a computer system.

Naming rules and naming attributes

See "Naming rules and naming attributes " on page 78 for the naming rules and naming attributes of each class shown in Figure 25.

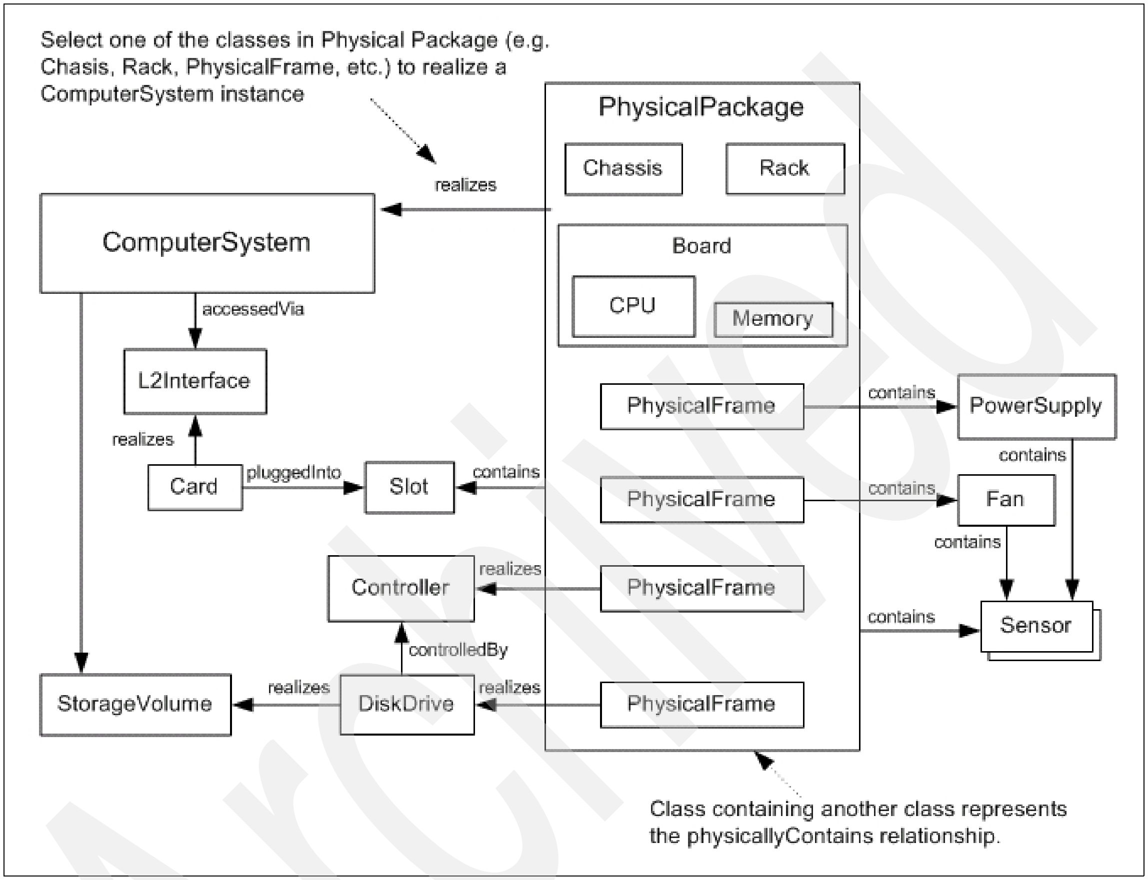

Figure 25 Instance diagram, server hardware

Important relationships

Table 28 shows the important relationships in this scenario.

Table 28 Important relationships, server hardware

Source class	Relationship type	Target class	Cardinality
Board	physicallyContains	CPU	1:m
Board	physicallyContains	Memory	1:m
Card	pluggedInto	Slot	1:1

Source class	Relationship type	Target class	Cardinality
Card	realizes	L2Interface	1:1
ComputerSystem	accessedVia	L2Interface	1:m
ComputerSystem	contains	StorageVolume	1:m
ComputerSystem	runsOn	PhysicalPackage	1:1
DiskDrive	controlledBy	Controller	m:n
DiskDrive	realizes	StorageVolume	m:n
Fan	contains	Sensor	1:m
PhysicalFrame	contains	Fan	1:m
PhysicalFrame	contains	PowerSupply	1:m
PhysicalFrame	realizes	Controller	1:1
PhysicalFrame	realizes	DiskDrive	1:1
PhysicalPackage	contains	Sensor	1:m
PhysicalPackage	contains	Slot	1:m
PhysicalPackage	physicallyContains	Board	1:m
PhysicalPackage	physicallyContains	Chassis	1:m
PhysicalPackage	physicallyContains	PhysicalFrame	1:m
PhysicalPackage	physicallyContains	Rack	1:m
PhysicalPackage	realizes	ComputerSystem	1:1
PowerSupply	contains	Sensor	1:m

Potential questions

Here are some related questions:

- What type of computer system is this?
- What CPUs are on this computer system (how many) and what are their clock speeds?
- How much memory does this computer system have?

- What is the bandwidth of this computer system?
- What is the primary MAC address of this computer system?
- What are the other MAC addresses of this computer system?
- How much disk space is on this computer system?
- Does this computer system have a redundant power supply?
- What type of power supply does this computer system contain?
- What type of fan does this computer system contain?
- How many expansion slots does this system have?

Blade enclosure

This scenario shows the physical/hardware aspects of a blade enclosure. This shows a basic blade enclosure configuration containing a single blade that is easily adaptable to the actual number of blades (or other servers) that are contained. The specific operating system classes would be determined by the actual operating system installation.

Naming rules and naming attributes

See "Naming rules and naming attributes " on page 78 for the naming rules and naming attributes of each class shown in Figure 26.

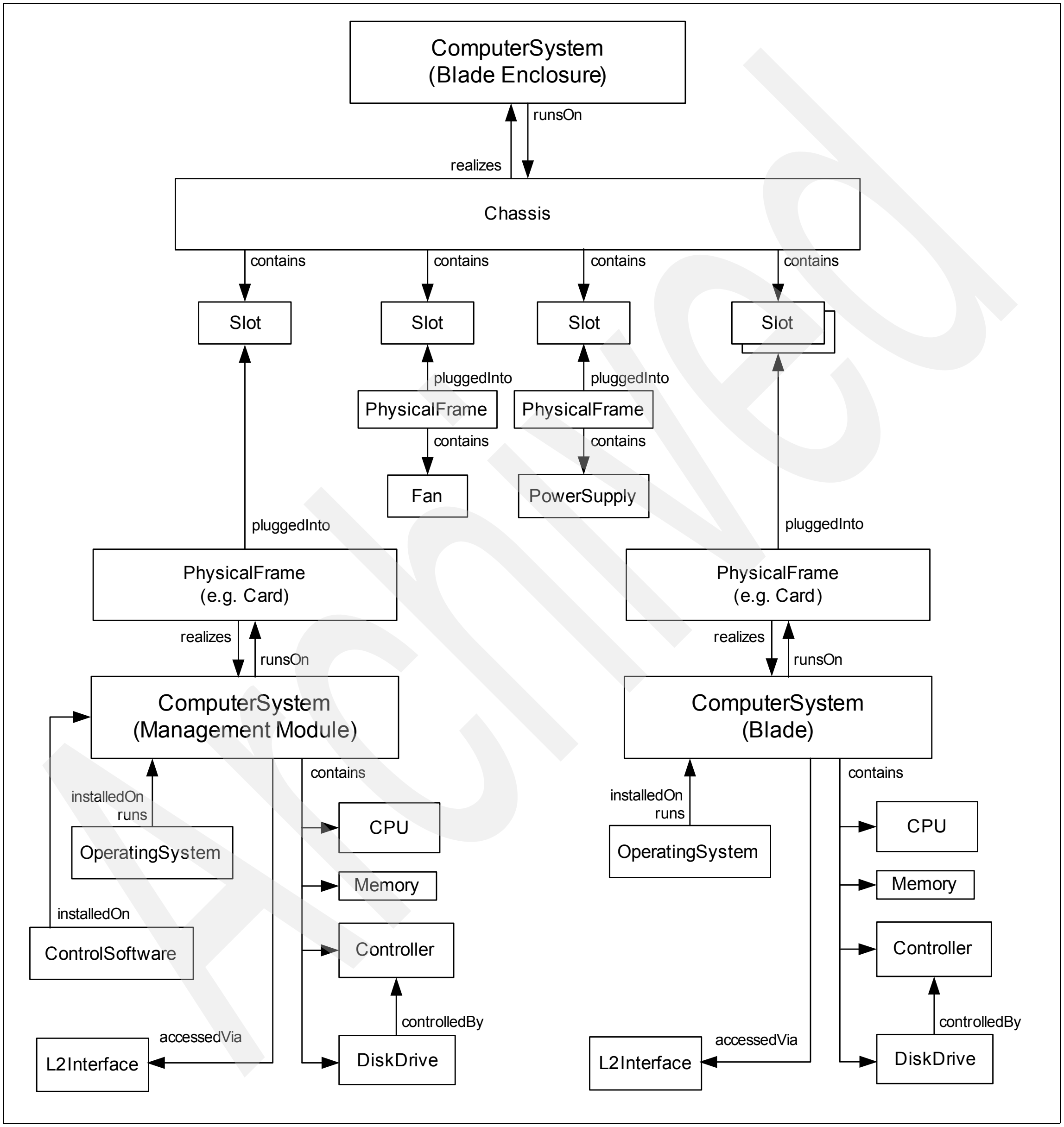

Figure 26 Instance diagram, blade enclosure

Important relationships

Table 29 shows the important relationships in this scenario.

Table 29 Important relationships, blade enclosure

Source class	Relationship type	Target class	Cardinality
Chassis	contains	Slot	1:m
Chassis	realizes	ComputerSystem	1:1
ComputerSystem	accessedVia	L2Interface	1:m
ComputerSystem	contains	DiskDrive	1:m
ComputerSystem	contains	Controller	1:m
ComputerSystem	contains	CPU	1:m
ComputerSystem	contains	Memory	1:1
ComputerSystem	runsOn	Chassis	1:1
ComputerSystem	runsOn	PhysicalFrame	1:1
ControlSoftware	installedOn	ComputerSystem	1:1
DiskDrive	controlledBy	Controller	m:n
OperatingSystem	installedOn	ComputerSystem	m:1
OperatingSystem	runsOn	ComputerSystem	1:1
PhysicalFrame	contains	Fan	1:m
PhysicalFrame	contains	PowerSupply	1:m
PhysicalFrame	pluggedInto	Slot	1:1
PhysicalFrame	realizes	ComputerSystem	1:1

Usage or implementation notes

The RelativePosition attribute of Slot can be used to determine whether a slot is at the front or the back of a blade enclosure.

Potential questions

Here are some related questions:

- What operating system is on a particular blade computer?
- What blades are in a particular blade enclosure?
- Which blade is in a particular slot?
- Which slots on the blade enclosure are available?
- Which blades are in the same blade enclosure?
- What is the total disk space on a particular blade enclosure?
- How many CPUs are available on a particular blade enclosure? What are the associated clock speeds?

Naming rules and naming attributes

This section lists the naming rules and naming attributes of all the classes used in the example scenarios. Table 30 also does the following:

- Appends an asterisk (*) to the name of a class to indicate that the class is a configuration item
- Provides the IdML element name of the class

Table 30 Naming rules and naming attributes for classes used in scenarios

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) **(IdML element name)**	**Naming rules**	**Superior (naming context)**
Activity (process.Activity)	0 = ActivityName, but Namespace and Owner are not provided 1 = superior, ActivityName	An instance representing the OrganizationalEntity that owns the Activity

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
AddressSpace * (sys.zOS.AddressSpace)	0 = superior1, JobName 1 = ManagedSystemName 2 = superior2, PID	superior1: An instance representing the zOS on which the AddressSpace runsOn superior2: An instance representing the OperatingSystem on which the AddressSpace runsOn
AppDescriptor (app.AppDescriptor)	0 = superior, Content	An instance representing the AppServer that contains the AppDescriptor
AppServer * (app.AppServer)	0 = superior, KeyName 1 = ManagedSystemName	An instance representing the BindAddress at which the AppServer is accessedVia
BindAddress (net.BindAddress)	0 = superior, Path, IpAddress, PortNumber	An instance representing the IpAddress that the BindAddress bindsAsPrimary
Board (phys.physpkg.Board)	0 = superior, Name, RelativePosition 1 = ManagedSystemName 2 = Model, SerialNumber, Manufacturer 3 = SystemBoardUUID	An instance representing the PhysicalPackage that physicallyContains the Board
BusinessProcess * (process.BusinessProcess)	0 = ActivityName, but Namespace and Owner are not provided 1 = superior, ActivityName	An instance representing the OrganizationalEntity that owns the BusinessProcess

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
BusinessSystem * (sys.BusinessSystem)	0 = Name 1 = superior, Name 2 = ManagedSystemName	An instance representing the ITSystem that contains the BusinessSystem
Card (phys.physpkg.Card)	0 = superior, Name, RelativePosition 1 = ManagedSystemName 2 = Model, SerialNumber, Manufacturer 3 = SystemBoardUUID	An instance representing the PhysicalPackage that physicallyContains the Card
Chassis (phys.physpkg.Chassis)	0 = ChassisUUID 1 = superior, Name, RelativePosition 2 = ManagedSystemName 3 = Model, SerialNumber, Manufacturer 4 = SystemBoardUUID	An instance representing the PhysicalPackage that physicallyContains the Chassis
CICSFile (sys.zOS.CICSFile)	0 = superior, DDName 1 = URI	0 = superior, DDName 1 = URI
CICSProgram (sys.zOS.CICSProgram)	0 = superior, Name	An instance representing the CICSRegion that contains the CICSProgram
CICSRegion * (sys.zOS.CICSRegion)	0 = NetID, AppID 1 = superior1, JobName 2 = superior2, KeyName 3 = ManagedSystemName	superior1: An instance representing the zOS on which the CICSRegion runsOn superior2: An instance representing the BindAddress at which the CICSRegion is accessedVia

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
CICSTransaction (sys.zOS.CICSTransaction)	0 = superior, Name	An instance representing the CICSRegion that contains the CICSTransaction
ComputerSystem* (sys.ComputerSystem)	0 = Signature 1 = Manufacturer, SerialNumber, Model, but VMID is not provided 2 = SystemBoardUUID 3 = PrimaryMACAddress 4 = HostSystem, VMID 5 = ManagedSystemName 6 = VMID, Manufacturer, SerialNumber, Model	
Controller (dev.Controller)	0 = superior, Name 1 = ManagedSystemName	An instance representing the ComputerSystem that contains the Controller
ControlSoftware (sys.ControlSoftware)	0 = superior, Name	An instance representing the ComputerSystem on which the ControlSoftware is installedOn
CPU * (sys.CPU)	0 = superior1 1 = superior2, IdentifyingNumber	superior1: An instance representing the ComputerSystem that contains the CPU superior2: An instance representing the Board that physicallyContains the CPU

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
Db2AdminServer * (app.db.db2.Db2AdminServer)	1 = superior, KeyName 0 = ManagedSystemName	An instance representing the BindAddress at which the Db2AdminServer is accessedVia
Db2BufferPool (db.db2.Db2BufferPool)	0 = superior, Name 1 = ManagedSystemName	An instance representing the Db2Database that contains the Db2BufferPool
Db2Container (app.db.db2.Db2Container)	0 = superior, Name 1 = ManagedSystemName	An instance representing the Db2TableSpace that contains the Db2Container
Db2Database (app.db.db2.Db2Database)	0 = superior1, Name, Alias 1 = ManagedSystemName 2 = superior2, Name 3 = superior3, Name	superior1: An instance representing the Db2Instance that contains the Db2Database superior2: An instance representing the DB2Subsystem that contains the Db2Database superior3: An instance representing the DB2DataSharingGroup that contains the Db2Database

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
DB2DataSharingGroup * (sys.zOS.DB2DataSharingGroup)	0 = superior1, Name 1 = ManagedSystemName 2 = superior2, SystemSpecificName 3 = Name, but Active is not provided	superior1: An instance representing the Sysplex that contains the DB2DataSharingGroup superior2: An instance representing the Sysplex that hosts the DB2DataSharingGroup
Db2Instance * (app.db.db2.Db2Instance)	1 = superior, KeyName 0 = ManagedSystemName	An instance representing the BindAddress at which the Db2Instance is accessedVia
Db2Schema (app.db.db2.Db2Schema)	0 = superior, Name 1 = ManagedSystemName	An instance representing the Db2Database that contains the Db2Schema
Db2Server * (app.db.db2.Db2Server)	0 = superior, KeyName 1 = ManagedSystemName	An instance representing the BindAddress at which the Db2Server is accessedVia
DB2Subsystem * (sys.zDB2Subsystem)	0 = superior1, SubsystemName 1 = superior2, KeyName 2 = ManagedSystemName	superior1: An instance representing the zOS that hosts the DB2Subsystem superior2: An instance representing the BindAddress at which the DB2Subsystem is accessedVia

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
Db2System (app.db.db2.Db2System)	0 = superior 1 = ManagedSystemName	An instance representing the ComputerSystem on which the Db2System runsOn
Db2TableSpace (app.db.db2.Db2TableSpac e)	0 = superior, Name 1 = ManagedSystemName	An instance representing the Db2Database that contains the Db2TableSpace
DiskDrive (dev.DiskDrive)	0 = AnsiT10Id 1 = superior, Name 2 = ManagedSystemName	An instance representing the ComputerSystem that contains the DiskDrive
DiskPartition (dev.DiskPartition)	0 = superior, Name 1 = ManagedSystemName	An instance representing the ComputerSystem that contains the DiskPartition
Document (process.Document)	0 = URI	
Fabric (storage.Fabric)	0 = Name 1 = WorldWideName 2 = ManagedSystemName	
Fan (phys.physpkg.Fan)	0 = superior, Name, RelativePosition 1 = ManagedSystemName	An instance representing the PhysicalPackage that contains the Fan
FCPort (dev.FCPort)	0 = superior, DeviceID 1 = PermanentAddress 2 = ManagedSystemName	An instance representing the ComputerSystem that contains the FCPort

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
FCSwitch * (storage.FCSwitch)	0 = WorldWideName 1 = Signature 2 = Manufacturer, SerialNumber, Model, but VMID is not provided 3 = SystemBoardUUID 4 = PrimaryMACAddress 5 = HostSystem, VMID 6 = ManagedSystemName 7 = VMID, Manufacturer, SerialNumber, Model	
FCVolume (dev.FCVolume)	0 = superior, Name 1 = ManagedSystemName	An instance representing the ComputerSystem that contains the FileSystem
FileSystem (sys.FileSystem)	0 = superior, MountPoint 1 = ManagedSystemName	An instance representing the ComputerSystem that contains the FileSystem
Firewall (net.Firewall)	0 = superior, Name	An instance representing the ComputerSystem that provides the Firewall
Fqdn (net.Fqdn)	0 = Fqdn	
HostBusAdapter * (storage.HostBusAdapter)	0 = WorldWideName	
HpUx * (sys.hpux.HpUx)	0 = superior, Name 1 = superior, OsId 2 = SystemGuid 3 = superior, OSName 4 = ManagedSystemName 5 = FQDN	An instance representing the ComputerSystem on which the HpUx is installedOn

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
IBMHTTPServer	0 = superior, KeyName 1 = ManagedSystemName	An instance representing the BindAddress at which the IBMHTTPServer accessedVia
IMSDatabase (sys.zOS.IMSDatabase)	0 = superior, Name	An instance representing the IMSSubsystem that contains the IMSDatabase
IMSProgram (sys.zOS.IMSProgram)	0 = superior, Name	An instance representing the IMSSubsystem that contains the IMSProgram
IMSSubsystem * (sys.zOS.IMSSubsystem)	0 = superior1, SubsystemName 1 = superior2, KeyName 2 = ManagedSystemName	superior1: An instance representing the zOS that hosts the IMSSubsystem superior2: An instance representing the BindAddress at which the IMSSubsystem is accessedVia
IMSTransaction (sys.zOS.IMSTransaction)	0 = superior, Name	An instance representing the IMSSubsystem that contains the IMSTransaction
IpAddress (net.IpAddress)	0 = DotNotation, but not AddressSpace 1 = DotNotation, AddressSpace	An instance representing the ComputerSystem that contains the IpInterface

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) **(IdML element name)**	**Naming rules**	**Superior (naming context)**
IpRoute (net.IpRoute)	0 = superior, Destination, NextHop	An instance representing the Router that manages the IpRoute
IpV4Address (net.IpV4Address)	0 = DotNotation, but not AddressSpace 1 = DotNotation, AddressSpace	
L2Interface (net.L2Interface)	0 = superior, Index 1 = superior, Name 2 = superior, CdpRef 3 = ManagedSystemName	An instance representing the ComputerSystem that is accessedVia via the L2Interface
Memory * (sys.Memory)	0 = superior	An instance representing the ComputerSystem that contains the Memory
MQReceiverChannel * (app.messaging.mq. MQReceiverChannel)	0 = superior, Name	An instance representing the MQQueueManager that federates the MQReceiverChannel
MQSenderChannel * (app.messaging.mq. MQSenderChannel)	0 = superior, Name	An instance representing the MQQueueManager that federates the MQReceiverChannel

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) **(IdML element name)**	**Naming rules**	**Superior (naming context)**
MQSubsystem * (sys.zOS.MQSubsystem).	0 = superior1, SubsystemName 1 = superior2, Name 2 = superior3, KeyName 3 = ManagedSystemName	superior1: An instance representing the zOS that hosts the MQSubsystem superior2: An instance representing the MQInstallation that contains the MQSubsystem superior3: An instance representing the BindAddress at which the MQSubsystem is accessedVia
OperatingSystem * (sys.OperatingSystem)	0 = superior, Name 1 = superior,OsId 2 = SystemGuid 3 = superior,OSName 4 = ManagedSystemName 5 = FQDN	An instance representing the ComputerSystem on which the OperatingSystem is installedOn
OracleBackgroundProcess (app.db.oracle. OracleBackgroundProcess)	0 = superior, Name 1 = ManagedSystemName	An instance representing the OracleInstance that contains the OracleBackgroundProcess
OracleControlFile (app.db.oracle.OracleControlFile)	0 = superior, Name 1 = ManagedSystemName	An instance representing the OracleDatabase that contains the OracleControlFile

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
OracleDatabase (app.db.oracle.OracleDatabase)	0 = superior, Name 1 = DomainName, Name 2 = ManagedSystemName	An instance representing the OracleInstance that provides the OracleDatabase
OracleDataFile (app.db.oracle.OracleDataFile)	0 = superior, Name 1 = ManagedSystemName	An instance representing the OracleDatabase that contains the OracleDataFile
OracleDBLink (app.db.oracle.OracleDBLink)	0 = superior, ServiceName 1 = ManagedSystemName	An instance representing the OracleDatabase that contains the OracleDBLink
OracleInitValue (app.db.oracle.OracleInitValue)	0 = superior, Name	An instance representing the OracleDatabase that contains the OracleInitValue
OracleInstance * (app.db.oracle.OracleInstance)	0 = superior1, SID, Home 1 = Hostname, SID, Home 2 = superior2, KeyName 3 = ManagedSystemName	superior1: An instance representing the ComputerSystem on which the OracleInstance runsOn superior2: An instance representing the BindAddress at which the OracleInstance is accessedVia
OracleListener (app.db.oracle.OracleListener)	0 = superior, Name 1 = ManagedSystemName	An instance representing the OracleServer that contains the OracleListener

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
OracleRedoLogFile (app.db.oracle.OracleRedoLogFile)	0 = superior, Name 1 = ManagedSystemName	An instance representing the OracleDatabase that contains the OracleRedoLogFile
OracleSchema (app.db.oracle.OracleSchema)	0 = superior, Name 1 = ManagedSystemName	An instance representing the OracleDatabase that contains the OracleSchema
OracleServer (app.db.oracle.OracleServer)	0 = ConfigFile, Host 1 = ManagedSystemName	
OracleServerProcess (app.db.oracle. OracleServerProcess)	0 = superior, PID 1 = ManagedSystemName	An instance representing the OperatingSystem on which the OracleServerProcess runsOn
OracleSGAValue (app.db.oracle.OracleSGAValue)	0 = superior, Name	An instance representing the OracleInstance that contains the OracleSGAValue
OracleTableSpace (app.db.oracle.OracleTableSpace)	0 = superior, Name 1 = ManagedSystemName	An instance representing the OracleDatabase that contains the OracleTableSpace
Organization (process.Organization)	0 = GlobalName 1 = superior, EntityName	An instance representing the OrganizationalEntity that manages the Organization

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
OrganizationalEntity (process.OrganizationalEntity)	0 = GlobalName 1 = superior, EntityName	An instance representing the OrganizationalEntity that manages the OrganizationalEntity
PhysicalFrame (phys.physpkg.PhysicalFrame)	0 = superior, Name, RelativePosition 1 = ManagedSystemName 2 = Model, SerialNumber, Manufacturer 3 = SystemBoardUUID	An instance representing the PhysicalPackage that physicallyContains the PhysicalFrame
PhysicalPackage (phys.physpkg.PhysicalPackage)	0 = Model, SerialNumber, Manufacturer 1 = SystemBoardUUID 2 = ManagedSystemName	
PowerSupply (phys.physpkg.PowerSupply)	0 = superior, Name, RelativePosition 1 = ManagedSystemName	An instance representing the PhysicalPackage that contains the PowerSupply
Rack (phys.physpkg.Rack)	0 = superior, Name, RelativePosition 1 = ManagedSystemName 2 = Model, SerialNumber, Manufacturer 3 = SystemBoardUUID	An instance representing the PhysicalPackage that physicallyContains the Rack
Router (net.Router)	0 = superior, Name	An instance representing the ComputerSystem that provides the Router
RuntimeProcess (sys.RuntimeProcess)	0 = superior, PID 1 = ManagedSystemName	An instance representing the OperatingSystem on which the RuntimeProcess runsOn

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
Sensor (phys.physpkg.Sensor)	0 = superior, RelativePosition, Name 1 = ManagedSystemName	An instance representing the PhysicalPackage that contains the Sensor
Slot (phys.physconn.Slot)	0 = superior, Name, RelativePosition 1 = ManagedSystemName	An instance representing the PhysicalPackage that contains the Slot
SoftwareInstallation* (app.SoftwareInstallation)	0 = superior, ManufacturerName, ProductName, InstalledLocation 1 = superior, ProductId, ManufacturerName, InstalledLocation 2 = superior, ManufacturerName, ProductName, but not InstalledLocation 3 = ManagedSystemName	An instance representing the OperatingSystem on which the SoftwareInstallation is installedOn
SqlServer * (app.db.mssql.SqlServer)	0 = superior, KeyName 1 = ManagedSystemName	An instance representing the BindAddress at which the SqlServer is accessedVia
SqlServerConfig (app.db.mssql.SqlServerConfig)	0 = superior, ConfigId	An instance representing the SqlServer that is configuredUsing the SqlServerConfig
SqlServerDatabase (app.db.mssql.SqlServerDatabase)	0 = superior, Name 1 = ManagedSystemName	An instance representing the SqlServer that contains the SqlServerDatabase

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
SqlServerProcess (app.db.mssql.SqlServerProcess)	0 = superior, Spid 1 = ManagedSystemName	An instance representing the SqlServer that contains the SqlServerProcess
SqlServerTable (app.db.mssql.SqlServerTable)	0 = superior, Name 1 = ManagedSystemName	An instance representing the SqlServerDatabase that contains the SqlServerTable
StorageExtent (dev.StorageExtent)	0 = superior, Name 1 = ManagedSystemName	An instance representing the ComputerSystem that contains the StorageExtent
StoragePool * (storage.StoragePool)	0 = AnsiT10Id 1 = ManagedSystemName	
StorageSubSystem * (storage.StorageSubSystem)	0 = AnsiT10Id 1 = Signature 2 = Manufacturer, SerialNumber, Model, but VMID is not provided 3 = SystemBoardUUID 4 = PrimaryMACAddress 5 = HostSystem, VMID 6 = ManagedSystemName 7 = VMID, Manufacturer, SerialNumber, Model	
StorageVolume (dev.StorageVolume)	0 = superior, Name 1 = ManagedSystemName	An instance representing the ComputerSystem that contains the StorageVolume
TcpPort (net.TcpPort)	0 = superior, PortNumber 1 = ManagedSystemName	An instance representing the IpInterface to which the TcpPort bindsTo

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
UdpPort (net.UdpPort)	0 = superior, PortNumber 1 = ManagedSystemName	An instance representing the IpInterface to which the UdpPort bindsTo
Unix * (sys.Unix)	0 = superior, Name 1 = superior,OsId 2 = SystemGuid 3 = superior,OSName 4 = ManagedSystemName 5 = FQDN	An instance representing the ComputerSystem on which the Unix is installedOn
UnixProcess (sys.unix.UnixProcess)	0 = superior, PID 1 = ManagedSystemName	An instance representing the OperatingSystem on which the UnixProcess runsOn
Vip (net.vip.Vip)	0 = superior, VipAddress	An instance representing the VipFunction that provides the Vip
WebLogicCluster (app.j2ee.weblogic. WebLogicCluster)	0 = superior1, Name 1 = superior2, Port 2 = ManagedSystemName	superior1: An instance representing the WebLogicDomain that federates the WebLogicCluster superior2: An instance representing the IpAddress at which the WebLogicCluster is accessedVia
WebLogicDomain (app.j2ee.weblogic. WebLogicDomain)	0 = superior 1 = Owner, Name 2 = ManagedSystemName	An instance representing the BindAddress at which the WebLogicDomain is accessedVia

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
WebLogicEJBContainer (app.j2ee.weblogic. WebLogicEJBContainer)	0 = superior, Name 1 = ManagedSystemName	An instance representing the AppServer that contains the WebLogicEJBContainer
WebLogicEJBModule * (app.j2ee.weblogic. WebLogicEJBModule)	0 = superior1, Name, FileName 1 = superior2, Name, FileName 2 = ManagedSystemName	superior1: An instance representing the J2EEDomain to which the WebLogicEJBModule is deployedTo superior2: An instance representing the AppServer to which the WebLogicEJBModule is deployedTo
WebLogicJ2EEApplication * (app.j2ee.weblogic. WebLogicJ2EEApplication)	0 = superior1, Name, FileName 1 = superior2, Name, FileName 2 = ManagedSystemName	superior1: An instance representing the J2EEDomain to which the WebLogicJ2EEApplicati on is deployedTo superior2: An instance representing the AppServer to which the WebLogicJ2EEApplicati on is deployedTo

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
WebLogicJDBCDataSource * (app.j2ee.weblogic. WebLogicJDBCDataSource)	0 = superior1, Name 1 = superior2, Name 2 = superior1, JNDIName 3 = superior2, JNDIName 4 = ManagedSystemName	superior1: An instance representing the J2EEDomain of which the WebLogicJDBCDataSource is a memberOf superior2: An instance representing the J2EEServer that uses the WebLogicJDBCDataSource
WebLogicJMSServer * (app.j2ee.weblogic. WebLogicJMSServer)	0 = superior1, Name 1 = superior2, Name 3 = ManagedSystemName	superior1: An instance representing the J2EEDomain of which the WebLogicJMSServer is a memberOf superior2: An instance representing the J2EEServer that uses the WebLogicJMSServer
WebLogicJTA (app.j2ee.weblogic.WebLogicJTA)	0 = superior 1 = ManagedSystemName	An instance representing the WebLogicDomain that contains the WebLogicJTA
WebLogicMachine (app.j2ee.weblogic. WebLogicMachine)	0 = superior, Name 1 = ManagedSystemName	An instance representing the WebLogicDomain of which the WebLogicMachine is a memberOf

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
WebLogicNodeManager (app.j2ee.weblogic. WebLogicNodeManager)	0 = superior 1 = ManagedSystemName	An instance representing the WebLogicMachine that contains the WebLogicNodeManager
WebLogicServer * (app.j2ee.weblogic. WebLogicServer)	0 = superior1, Name 1 = superior2, KeyName 2 = ManagedSystemName	superior1: An instance representing the J2EEDomain of which the WebLogicServer is a memberOf superior2: An instance representing the BindAddress at which the WebLogicServer is accessedVia
WebLogicSSLSettings (app.j2ee.weblogic. WebLogicSSLSettings)	0 = superior	An instance representing the WebLogicServer that is configuredUsing the WebLogicSSLSettings
WebLogicWebContainer (app.j2ee.weblogic. WebLogicWebContainer)	0 = superior, Name 1 = ManagedSystemName	An instance representing the AppServer that contains the WebLogicWebContainer

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
WebLogicWebModule * (app.j2ee.weblogic. WebLogicWebModule)	0 = superior1, Name, FileName 1 = superior2, Name, FileName 2 = ManagedSystemName	superior1: An instance representing the J2EEDomain to which the WebLogicWebModule is deployedTo superior2: An instance representing the AppServer to which the WebLogicWebModule is deployedTo
WebService * (soa.WebService)	0 = Namespace, Name 1 = PrimarySAP 2 = Name, Host 3 = ManagedSystemName	
WebSphereCell (app.j2ee.websphere. WebSphereCell)	0 = superior 1 = Owner, Name 2 = ManagedSystemName	An instance representing the BindAddress at which the WebSphereCell is accessedVia
WebSphereCluster (app.j2ee.websphere. WebSphereCluster)	0 = superior1, Name 1 = superior2, Port 2 = ManagedSystemName	superior1: An instance representing the WebSphereCell that federates the WebSphereCluster superior2: An instance representing the IpAddress at which the WebSphereCluster is accessedVia

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
WebSphereDeploymentManager * (app.j2ee.websphere.WebSphereDeploymentManager)	0 = superior1, Name 1 = AddressSpace 2 = superior2, Name 3 = superior3, KeyName 4 = ManagedSystemName	superior1: An instance representing the WebSphereNode that federates the WebSphereDeployment Manager superior2: An instance representing the J2EEDomain of which the WebSphereDeployment Manager is a memberOf superior3: An instance representing the BindAddress at which the WebSphereDeployment Manager is accessedVia
WebSphereEJBContainer (app.j2ee.websphere.WebSphereEJBContainer)	0 = superior, Name 1 = ManagedSystemName	An instance representing the AppServer that contains the WebSphereEJBContainer

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
WebSphereEJBModule * (app.j2ee.websphere. WebSphereEJBModule)	0 = superior1, Name, FileName 1 = superior2, Name, FileName 2 = ManagedSystemName	superior1: An instance representing the J2EEDomain to which the WebSphereEJBModule is deployedTo superior2: An instance representing the AppServer to which the WebSphereEJBModule is deployedTo
WebSphereGlobalSecuritySettings (app.j2ee.websphere. WebSphereGlobalSecuritySettings)	0 = superior	An instance representing the WebSphereCell that is configuredUsing the WebSphereGlobalSecuritySettings
WebSphereJ2EEApplication * (app.j2ee.websphere. WebSphereJ2EEApplication)	0 = superior1, Name, FileName 1 = superior2, Name, FileName 2 = ManagedSystemName	superior1: An instance representing the J2EEDomain to which the WebSphereJ2EEApplication is deployedTo superior2: An instance representing the AppServer to which the WebSphereJ2EEApplication is deployedTo

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
WebSphereJDBCDataSource * (app.j2ee.websphere.WebSphereJDBCDataSource)	0 = superior1, Name 1 = superior2, Name 3 = ManagedSystemName	superior1: An instance representing the J2EEDomain of which the WebSphereJDBCDataSource is a memberOf superior2: An instance representing the J2EEServer that uses the WebSphereJDBCDataSource
WebSphereJMSServer * (app.j2ee.websphere.WebSphereJMSServer)	0 = superior1, Name 1 = superior2, Name 2 = superior3, Name 3 = ManagedSystemName	superior1: An instance representing the J2EEDomain of which the WebSphereJMSServer is a memberOf superior2: An instance representing the J2EEServer that uses the WebSphereJMSServer superior3: An instance representing the WebSphereNode that provides the WebSphereJMSServer
WebSphereNamedEndpoint (app.j2ee.websphere.WebSphereNamedEndpoint)	0 = superior, Name	An instance representing the BindAddress at which the WebSphereNamedEndpoint is accessedVia

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) **(IdML element name)**	**Naming rules**	**Superior (naming context)**
WebSphereNode (app.j2ee.websphere. WebSphereNode)	0 = superior1 1 = superior2, Name 2 = ManagedSystemName	superior1: An instance representing the BindAddress at which the WebSphereNode is accessedVia superior2: An instance representing the WebSphereCell of which the WebSphereNode is a memberOf
WebSphereNodeAgent * (app.j2ee.websphere. WebSphereNodeAgent)	0 = superior1, Name 1 = AddressSpace 2 = superior2, Name 3 = superior3, KeyName 4 = ManagedSystemName	superior1: An instance representing the WebSphereNode that federates the WebSphereNodeAgent superior2: An instance representing the J2EEDomain of which the WebSphereNodeAgent is a memberOf superior3: An instance representing the BindAddress at which the WebSphereNodeAgent is accessedVia

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
WebSphereServer * (app.j2ee.websphere. WebSphereServer)	0 = superior1, Name 1 = AddressSpace 2 = superior2, Name 3 = superior3, KeyName 4 = ManagedSystemName	superior1: An instance representing the WebSphereNode that federates the WebSphereServer superior2: An instance representing the J2EEDomain of which the WebSphereServer is a memberOf superior3: An instance representing the BindAddress at which the WebSphereServer is accessedVia
WebSphereTransactionServ ice (app.j2ee.websphere. WebSphereTransactionServ ice)	0 = superior 1 = ManagedSystemName	An instance representing the WebSphereServer that provides the WebSphereTransaction Service
WebSphereUserRegistry (app.j2ee.websphere. WebSphereUserRegistry)	0 = superior, RegistryType	An instance representing the WebSphereGlobalSecu ritySettings that uses the WebSphereUserRegistr y
WebSphereVirtualHost (app.j2ee.websphere. WebSphereVirtualHost)	0 = superior, Name 1 = ManagedSystemName	An instance representing the WebSphereCell that provides the WebSphereVirtualHost

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
WebSphereWebContainer (app.j2ee.websphere. WebSphereWebContainer)	0 = superior, Name 1 = ManagedSystemName	An instance representing the AppServer that contains the WebSphereWebContai ner
WebSphereWebModule * (app.j2ee.websphere. WebSphereWebModule)	0 = superior1, Name, FileName 1 = superior2, Name, FileName 2 = ManagedSystemName	superior1: An instance representing the J2EEDomain to which the WebSphereWebModule is deployedTo superior2: An instance representing the AppServer to which the WebSphereWebModule is deployedTo
WindowsOperatingSystem * (sys.windows. WindowsOperatingSystem)	0 = superior, Name 1 = superior,OsId 2 = SystemGuid 3 = superior,OSName 4 = ManagedSystemName 5 = FQDN	An instance representing the ComputerSystem on which the WindowsOperatingSyst em is installedOn
WindowsService (sys.windows.WindowsServ ice)	0 = superior, Name 1 = ManagedSystemName	An instance representing the WindowsOperatingSyst em on which the WindowsService is installedOn
WSEndpoint (soa.WSEndpoint)	0 = Port, Provider 1 = BindAddress	

Class (* indicates that the class is a CI, meaning that it implements the configurationItem interface) (IdML element name)	Naming rules	Superior (naming context)
WSOperation (soa.WSOperation)	0 = Namespace, ActivityName 1 = ActivityName, but Namespace and Owner are not provided 2 = superior, ActivityName	An instance representing the OrganizationalEntity that owns the WSOperation
WSPort (soa.WSPort)	0 = Name, Namespace 1 = BindAddress	
WSPortType (soa.WSPortType)	0 = Namespace, InterfaceName 1 = InterfaceName, but Namespace is not provided	
ZLinux * (sys.zOS.ZLinux)	0 = superior, Name 1 = superior,OsId 2 = SystemGuid 3 = superior,OSName 4 = ManagedSystemName 5 = FQDN	An instance representing the ComputerSystem on which the ZLinux is installedOn
Zone * (storage.Zone)	0 = Active, Name 1 = ManagedSystemName	
ZoneSet (storage.ZoneSet)	0 = Name, Active	
ZOS * (sys.zOS.ZOS)	0 = NetidSSCP 1 = superior1, SMFID 2 = superior2, Name 3 = superior2, OsId 4 = SystemGuid 5 = superior2,OSName 6 = ManagedSystemName 7 = FQDN	superior1: OrganizationalEntity that owns the zOS superior2: ComputerSystem on which the zOS is installedOn

Sample model objects

This section provides sample code segment for constructing model objects for some classes used in this document.

Example 1 Sample IdML

```
Computer System
ComputerSystem aComputerSystem =
 (ComputerSystem) ModelObjectFactory.newInstance(ComputerSystem.class);
aComputerSystem.setLabel(“lab135009”);
aComputerSystem.setSourceToken(“lab135009-linux”);
aComputerSystem.setCPUSpeed(1400000000);
aComputerSystem.setSignature(“9.87.135.9(00096B9A3C43)”);
aComputerSystem.setCPUType(“pentiumIII ");
aComputerSystem.setNumCPUs(1);
aComputerSystem.setFqdn(“lab135009”);
aComputerSystem.setManufacturer("IBM");
aComputerSystem.setMemorySize(2074976);
aComputerSystem.setModel(“xSeries 330 -[867443X]");
aComputerSystem.setSerialNumber(“KBTY759”);
aComputerSystem.setPrimaryMACAddress(“00096B9A3C43”);

Operating System
OperatingSystem anOperatingSystem =
          (OperatingSystem)ModelObjectFactory.newInstance(OperatingSys-
tem.class);

anOperatingSystem.setSystemGuid("EB7F8B18-1DD1-11B2-AA58-EA5EFEB B8D7D
");

anOperatingSystem.setLabel("lab135009-linux LINUX 2.6.9-5.ELsmp ");

anOperatingSystem.setSourceToken(“OS-lab135009-linux ”);

anOperatingSystem.setBootTime(1169574130000);

anOperatingSystem.setRelease("6");

anOperatingSystem.setOsId(“1”);

anOperatingSystem.setOSMajorVersion(“2");

anOperatingSystem.setOSName(“LINUX ”);

anOperatingSystem.setOSVersion(“2.6.9-5.ELsmp”);
```

```
anOperatingSystem.setVirtualMemorySize(2031608);
```

Software Installation

```
SoftwareInstallation aSoftwareInstallation =
 (SoftwareInstallation)ModelObjectFactory.newInstance
 (SoftwareInstallation.class);
aSoftwareInstallation.setLabel("SoftwareInstallation: udev ");
aSoftwareInstallation.setInstalledLocation("/etc/,/sbin/,/usr/");
aSoftwareInstallation.setManufacturerName("Red Hat ");
aSoftwareInstallation.setRelease(10);
aSoftwareInstallation.setMajorVersion("039");
aSoftwareInstallation.setModifier(8);
aSoftwareInstallation.setProductName("udev ");
aSoftwareInstallation.setVersionString("039.10.8");
```

IP Interfaces

```
IpInterface anIpInterface =
 (IpInterface) ModelObjectFactory.newInstance(IpInterface.class);
anIpInterface.setSourceToken("IP-9.87.135.9");
```

Fqdn

```
Fqdn aFqdn =
 (Fqdn) ModelObjectFactory.newInstance(Fqdn.class);
aFqdn.setLabel("lab135009");
aFqdn.setFqdn("lab135009");
```

IP Address

```
IpAddress anIpAddress =
 (IpAddress) ModelObjectFactory.newInstance(IpAddress.class);
anIpAddress.setSourceToken("9.87.135.9");
```

```
anIpAddress.setLabel("9.87.135.9");
anIpAddress.setDotNotation("9.87.135.9" );
```

The code segment in Example 2 builds a RunsOn relationship between one source object and a target object. In building a relationship model object, only the GUID of the source or target object needs to be set in the source or the target model object. However, a relationship model object must be built for each relationship type. That is, one for RunsOn, one for Federates, one for InstalledOn, and so on.

Example 2 Relationships

```
RunsOn aRelationship =
   (RunsOn) ModelObjectFactory.newInstance(RunsOn.class);
ModelObject sourceModelObj =
   (ModelObject) ModelObjectFactory.newInstance(ModelObject.class);
ModelObject targetModelObj =
   (ModelObject) ModelObjectFactory.newInstance(ModelObject.class);

sourceModelObj.setGuid(guid of the source object);
targetModelObj.setGuid(guid of the target object);
aRelationship.setSource(sourceModelObj);
aRelationship.setType("RunsOn");
aRelationship.setTarget(targetModelObj);
```

Example 3 is a sample IdML code.

Example 3 Sample IdML code

```
Computer System
<cdm:sys.ComputerSystem id="3987" sourceToken="lab135009-linux">
<cdm:Type>ComputerSystem</cdm:Type>
<cdm:CPUType>pentiumIII</cdm:CPUType>
<cdm:Fqdn>lab135009</cdm:Fqdn>
<cdm:Label>lab135009</cdm:Label>
<cdm:Signature>9.87.135.9(00096B9A3C43)</cdm:Signature>
<cdm:SerialNumber>KBTY759</cdm:SerialNumber>
<cdm:Manufacturer>IBM</cdm:Manufacturer>
<cdm:MemorySize>2074976</cdm:MemorySize>
<cdm:NumCPUs>1</cdm:NumCPUs>
<cdm:PrimaryMACAddress>00096B9A3C43</cdm:PrimaryMACAddress>
<cdm:CPUSpeed>1400000000</cdm:CPUSpeed>
<cdm:Model>xSeries 330 -[867443X]-</cdm:Model>
</cdm:sys.ComputerSystem>
Operating System
```

```
  <cdm:sys.OperatingSystem id="3988" sourceToken="OS-lab135009-linux">
       <cdm:OSVersion>2.6.9-5.ELsmp</cdm:OSVersion>
       <cdm:OsId>1</cdm:OsId>
       <cdm:SystemGuid>EB7F8B18-1DD1-11B2-AA58-EA5EFEBB8D7D
        </cdm:SystemGuid>
       <cdm:Release>6</cdm:Release>
       <cdm:Label>lab135009-linux LINUX 2.6.9-5.ELsmp</cdm:Label>
       <cdm:VirtualMemorySize>2031608</cdm:VirtualMemorySize>
       <cdm:OSName>LINUX</cdm:OSName>
       <cdm:MajorVersion>2</cdm:MajorVersion>
       <cdm:BootTime>1169574130000</cdm:BootTime>
  </cdm:sys.OperatingSystem>
IP Interface
     <cdm:net.IpInterface id="3989" sourceToken="IP-9.87.135.9" />
Fqdn
<cdm:net.Fqdn id="3990">
<cdm:Label>lab135009</cdm:Label>
<cdm:Fqdn>lab135009</cdm:Fqdn>
     </cdm:net.Fqdn>
IpV4Address
<cdm:net.IpV4Address id="3991" sourceToken="9.87.135.9">
<cdm:Label>9.87.135.9</cdm:Label>
<cdm:DotNotation>9.87.135.9</cdm:DotNotation>
</cdm:net.IpV4Address>
Software Installation
<cdm:app.SoftwareInstallation id="3994">

<cdm:InstalledLocation>/etc/,/sbin/,/usr/</cdm:InstalledLocation>
     <cdm:VersionString>039.10.8</cdm:VersionString>
     <cdm:ManufacturerName>Red Hat</cdm:ManufacturerName>
     <cdm:Release>10</cdm:Release>
     <cdm:Label>SoftwareInstallation: udev</cdm:Label>
     <cdm:ProductName>udev</cdm:ProductName>
     <cdm:Modifier>8</cdm:Modifier>
     <cdm:MajorVersion>039</cdm:MajorVersion>
</cdm:app.SoftwareInstallation>
Relationship
<cdm:installedOn source="3988" target="3987" />
<cdm:runsOn source="3988" target="3987" />
<cdm:contains source="3987" target="3989" />
<cdm:bindsTo source="3989" target="3991" />
<cdm:assignedTo source="3990" target="3991" />
<cdm:installedOn source="3994" target="3988" />
```

The team that wrote this IBM Redpaper

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Ling Tai is a Senior Software Engineer at IBM Tivoli Software group in Research Triangle Park, North Carolina. She was a design lead for the initial CMDB project that supports the Tivoli Common Data Model. Her areas of expertise include database management, BIRT reporting, Internet security, workflow management, and system and network management, and she has led projects in all these areas.

Ron Baker started his career in the aerospace industry, designing and writing numerical programs for engineering graphics and robotics applications. As relational databases began to appear, Ron was one of the early designers of their use in large-scale financial and configuration management systems at Boeing. After several years, he moved into research on database parallelism and integrity constraints at Amoco's Computing Research Center. Next he worked on the B-2 Stealth Bomber as an Engineering Configuration Database Specialist, where he addressed transitive closure problems like bill-of-material processing and reconciliation between configurations.

Ron joined IBM to work on object-oriented language integration with relational databases, and has worked on management products dealing with unstructured documents, search engines, and Internet services. He led the IBM Tivoli Common Data Model and identification work, along with the architecture of the ITIL-based Configuration Management Database (CMDB). He is currently a Senior Technical Staff Member in IBM Tivoli Software, where he is now responsible for a common Reporting Integtation initiative.

Elizabeth Edmiston received her Ph.D. in Computer Science from Duke University in 1989. Since then she has worked in academia teaching database and programming courses, and has worked in academic administration. She has led two successful accreditation projects. She is currently on the faculty at North Carolina Central University. Elizabeth specializes in data modeling and database design and has served as a consultant with a variety of organizations.

Ben Jeffcoat has been with IBM since 1997. He started as a user interface programmer with the Printing Systems Division in Boulder, Colorado. He moved on to work on developing Tivoli Systems Management products in 2000, first in Indianapolis, Indiana, and then in Research Triangle Park, North Carolina. His main technical interests include Java and Application Servers.

Notices

This document REDP-4389-00 was created or updated on February 11, 2008.

Send us your comments in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099, 2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Redpaper™

Trademarks

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

Redbooks (logo) ®	IBM®	Notes®
zSeries®	IMS™	Rational®
CICS®	Lotus Notes®	Tivoli®
DB2®	Lotus®	WebSphere®

The following terms are trademarks of other companies:

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

IT Infrastructure Library, IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

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.

EJB, Java, JDBC, J2EE, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

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