CA Spectrum®

Concepts Guide

Release 9.4

technologies

This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the “Documentation”) is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and CA governing your use of the CA software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CA.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

CA Technologies Product References

This document references the following CA Technologies products:

- CA Spectrum®
- CA Spectrum® Report Manager (Report Manager)
- CA Spectrum® Service Manager (Service Manager)
- CA Spectrum® Southbound Gateway Toolkit (Southbound Gateway)
- CA Spectrum® Modeling Gateway Toolkit (Modeling Gateway)
- CA Spectrum® Alarm Notification Manager (SANM)
- CA eHealth® (eHealth)
- CA Business Intelligence (CABI)

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: CA Spectrum Overview

This section contains the following topics:

About CA Spectrum

CA Spectrum is a services and infrastructure management system that monitors the state of managed elements including devices, applications, host systems, and connections. Status information such as fault and performance data from these elements is collected and stored. CA Spectrum constantly analyzes this information to track conditions within the computing infrastructure. If an abnormal condition is detected, it is isolated and you are alerted. CA Spectrum also presents you with possible causes and solutions to the problem.

The CA Spectrum design is based on the client/server model. Its primary server, the SpectroSERVER, is responsible for collecting, storing, and processing data. The SpectroSERVER uses Inductive Modeling Technology (IMT) to perform these functions. IMT combines an object-oriented database and the intelligence of inference handlers. The object-oriented database contains model types that define the representation of each managed element, and models that represent specific managed elements. The object-oriented database also contains relations that define possible associations between model types. The inference handlers provide more functionality to this system by reacting to events that CA Spectrum or managed elements produce.

The SpectroSERVER stores data in the knowledge base where model types, models, and relations are defined. The SpectroSERVER also polls managed elements and receives alert information from the computing infrastructure. The SpectroSERVER analyzes and stores this information in the knowledge base, and gives client applications access to this information.

CA Spectrum includes a number of client applications. Its main client application, OneClick, provides the graphical user interface that is used to monitor the network and launch other client applications. The views that are provided in the OneClick Console contain icons, tables, and graphs that represent the different elements of the network. These graphical components present status information and provide access to management facilities specific to the managed element they represent. All client application data is retrieved from the SpectroSERVER.

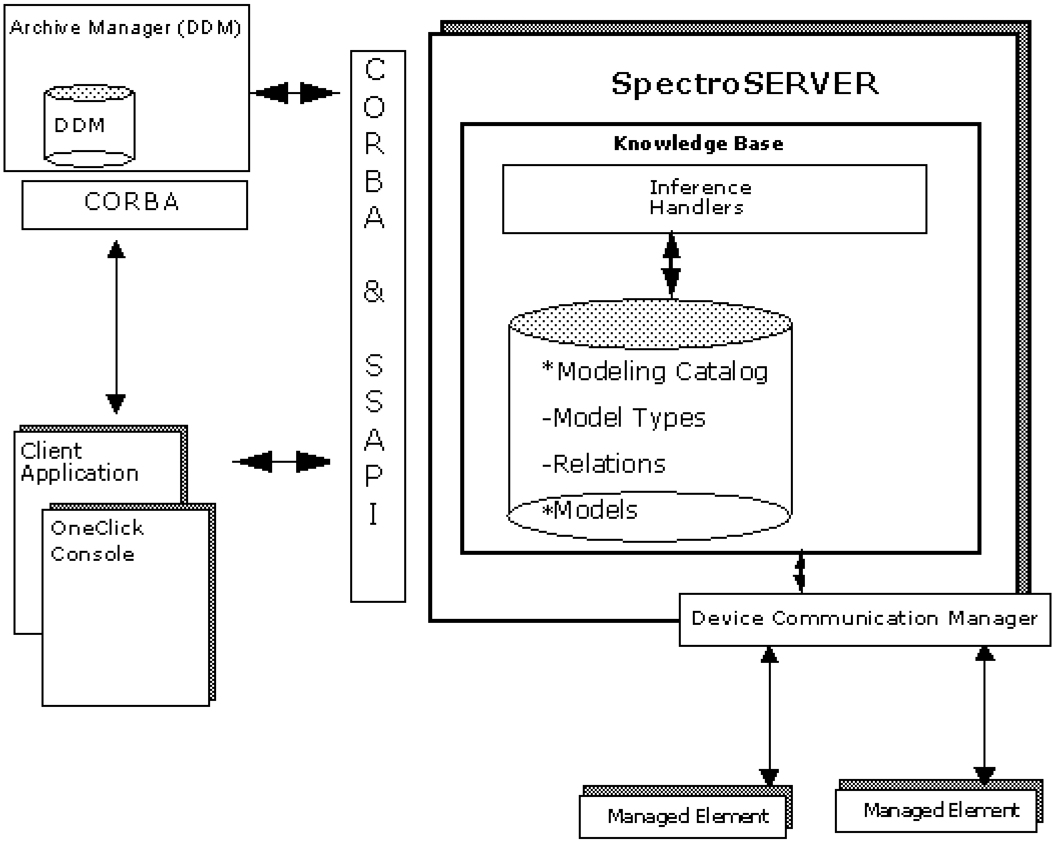
Chapter 2: SpectroSERVER and CA Spectrum Databases Overview

This section contains the following topics:

About the SpectroSERVER

The SpectroSERVER is the primary server for CA Spectrum, functioning as a database server, modeling engine, and device manager. The SpectroSERVER processes events, generates alarms, and tracks statistics for managed elements. This information is available to client applications and can be requested through the SpectroSERVER application programming interface (SSAPI) and the CA Spectrum CORBA interface.

The following illustration shows the SpectroSERVER components in a simplified view:



Note: The SpectroSERVER is also referred to as the VNM (Virtual Network Machine). Specifically, the term VNM refers to the portion of the SpectroSERVER that is responsible for modeling managed elements.

CA Spectrum Databases

CA Spectrum includes the following databases:

- SpectroSERVER database.

  For more information, see the *Database Management Guide*.

- Distributed Data Manager (DDM) database, which stores CA Spectrum events and statistical data for use across multiple landscapes.

  For more information, see the *Database Management Guide*.

- MIB Tools database, which supports the MIB Tools utility.

  For more information, see the *Certification User Guide*.

- Reporting database (see page 21), which support Report Manager and Service Manager.

  For more information, see the *Report Manager Installation and Administration Guide*.

- eHealth integration database, which the CA Spectrum and eHealth integration requires.

  For more information, see the *CA eHealth and CA Spectrum Integration and User Guide*.

Knowledge Base

The knowledge base is a main component of the SpectroSERVER. The knowledge base comprises both the data and the procedural information necessary to manage a computing infrastructure.

The knowledge base includes a component that stores model types, models, relations, and event and statistical information. A sophisticated system of models and relationships between models lets the knowledge base represent and store information about the network elements. This system of models and their relationships, when viewed as a single logical entity, describes the physical and logical topology of the computing infrastructure. CA Spectrum builds its *root cause analysis* capabilities on this foundation.

All models in the knowledge base are based on templates that are known as *model types*. The model types define the properties that make up an instantiated model. All model types are stored in the knowledge base modeling catalog.

The knowledge base also contains processes that give some intelligence to model types. These processes include inference handlers and actions. Process data is stored in memory while the SpectroSERVER is running and is also part of the knowledge base.

The knowledge base also uses the Archive Manager and the Distributed Data Manager (DDM) to store the historical event and statistical information about specific models. This information is accumulated over time, letting CA Spectrum gain extensive knowledge about the computing infrastructure being managed.

Archive Manager

Each landscape has an Archive Manager server that retrieves events and statistical data from the SpectroSERVER, compresses them, and stores them in the DDM database. Data compression enables storage of more performance data and decreases the network traffic between the applications and the DDM database.

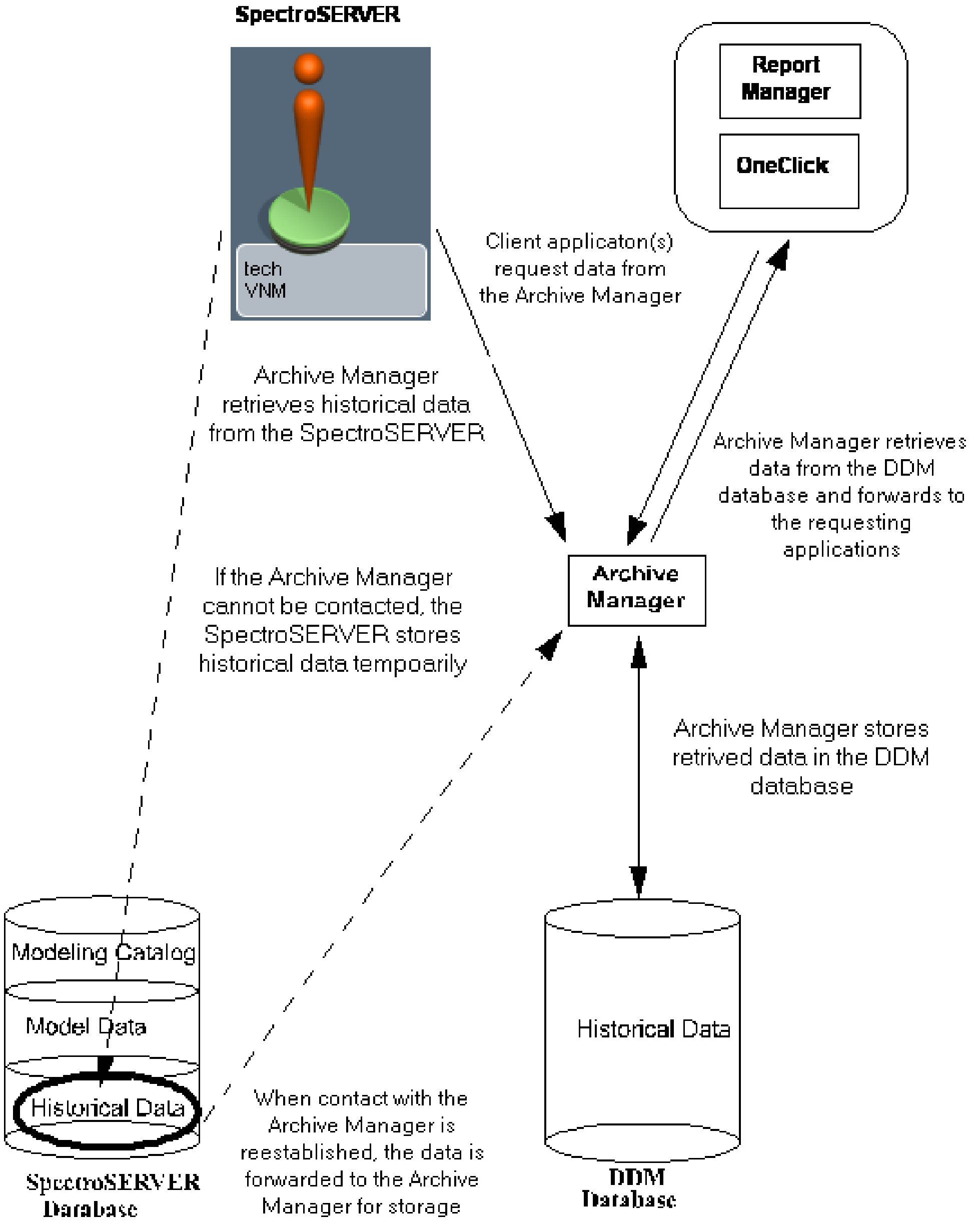
As the diagram shows, if the SpectroSERVER cannot contact the Archive Manager, the SpectroSERVER stores events and statistical data until contact is reestablished. The SpectroSERVER then sends the data to the Archive Manager for storage. The Events and Statistics Archive options in the .vnmrc file determine the amount of data that the SpectroSERVER stores. Options in the .configrc file determine the length of time that historical data is stored in the DDM database.

Note: For more information, see the *Distributed SpectroSERVER Administrator Guide.*

The Archive Manager can also provide the following information in response to requests from client applications:

- A list of landscapes for which information is available
- For each landscape, the time range of the available information and a list of model types for which information is available
- For each model type, a list of models for which information is available
- For each model, a list of attributes for which information is available
- Statistical data in the specified time range
- Event data in the specified time range

The following diagram shows the interaction between the SpectroSERVER and the DDM database.


SpectroSERVER
tech
VNM
Report Manager
OneClick
Client applicaton(s) request data from the Archive Manager
Archive Manager retrieves historical data from the SpectroSERVER
Archive Manager retrieves data from the DDM database and forwards to the requesting applications
If the Archive Manager cannot be contacted, the SpectroSERVER stores historical data tempoarily
Archive Manager
Archive Manager stores retrived data in the DDM database
Modeling Catalog
Model Data
Historical Data
SpectroSERVER Database
When contact with the Archive Manager is reestablished, the data is forwarded to the Archive Manager for storage
Historical Data
DDM Database

Modeling Catalog

The modeling catalog is the metadata repository of the knowledge base. The modeling catalog objects ship with CA Spectrum and are relatively static, but you can manipulate some catalog aspects for tuning purposes. You can also customize the modeling catalog to make CA Spectrum aware of new network technologies or new types of managed network elements. The following sections describe the various specific types of objects that are included in the modeling catalog.

Model Types

The model types correspond primarily to managed element families and are the templates that are used to build models. The model types contain the information, or attributes, required to manage a specific type of managed element. The model types possess the intelligence that tells CA Spectrum how the managed element represented by the model type behaves. This intelligence also describes how the managed element reacts to events occurring on the managed element or elsewhere in the network.

For example, the CA Spectrum modeling catalog contains a NokiaFW model type. This model type represents certain types of Nokia Firewalls such as IP330, IP440, IP650, and IP740. CA Spectrum uses it to create a model that represents a specific Nokia Firewall in a network.

Each model type is uniquely identified in the modeling catalog using a *model type handle* number, typically represented in hexadecimal format.

Model Type Attributes

Each model type has attributes that define the characteristics and properties of the managed element that the model type represents. These attributes can be either internal or external. The internal attributes reflect information that is specific to the CA Spectrum management of a particular element. The external attributes reflect objects from the MIBs that the managed element supports. All attributes have default values that are associated with the model type.

In many cases, attributes take on new values when a model of a specific model type is instantiated. The attribute values are specific to the managed element that the model represents. Some attributes, however, are shared attributes. All models of the given model type access the same shared attributes and their values. These attributes and values are not duplicated in memory or the database for each model.

Each attribute is uniquely identified in the knowledge base using a number that is known as an *attribute ID*, typically represented in hexadecimal format. Many attributes are used across numerous model types. For example, almost every model type in the modeling catalog uses the attribute Modeltype_name or IPAddress. The attribute IDs for these attributes remain the same across all model types. This normalization of attributes is achieved by using model type inheritance.

More information:

Model Type Hierarchy (see page 17)

Relations

Relations define the potential ways in which model types can be related to each other. Relations are defined in the CA Spectrum knowledge base. Examples of relations are Contains, Manages, and Connects_to. Each relation has a unique number, typically represented in hexadecimal format, that identifies it. This identifier is known as a *relation handle*.

More information:

Relation Descriptions (see page 49)

Meta-Rules

Meta-rules give meaning to a relation by defining the context in which the relation is used. A meta-rule identifies the model types that can participate in a relation. To understand the concept of a meta-rule, think of model types and relations as nouns and verbs, respectively. Noun and verb phrases are combined to form a sentence. For a sentence to be meaningful, it must meet three criteria:

- The sentence must be in the format (subject) noun + verb + (object) noun.
- The sentence must be logical; one cannot use any verb to link any two nouns.
- The sentence must reflect the reality.

The CA Spectrum notion of meta-rules enforces the second criterion. Meta-rules can be defined on a verb to limit the nouns that the verb can link. Define meta-rules carefully, so that the restrictions they impose on verbs are logical. Typically, several meta-rules govern each verb.

Consider a language where the following nouns and verbs are defined as the model types and relations for the objects constituting a network. Further, assume that meta-rules are defined to impose a logic on the way the nouns and verbs can be used together. Meta-rules are defined for a relation and consist of two model types: a left model type and a right model type. This left-right order in meta-rules is the format for building logical sentences. The left model type is the subject, the relation is the verb, and the right model type is the object of the sentence.

Nouns

building

room

network

LAN

printer

workstation

Verbs

contains

collects

Meta-rules

contains [building, room], [room, workstation]

collects [LAN, printer], [LAN, workstation], [network, LAN]

To create logical statements in this language, the first two criteria must be met. The following examples meet the first and second requirements and are realistic representations of a computing infrastructure:

- Engineering building contains testing lab
- Testing lab contains workstation ABC
- Engineering LAN collects workstation ABC
- Engineering LAN collects LaserJet printer

The following sentences are invalid because they do not use the noun/verb/noun format, and therefore do not meet the first criterion:

- Contains building collects
- Room LAN workstation

The following sentences meet the format requirement, but are either illogical or do not follow the defined meta-rules:

- Building contains workstation
- LAN collects room
- Printer collects LAN

Cardinality of Relations

Relations are defined to have a cardinality of either one-to-many or many-to-many. For example, the Contains relation has a one-to-many cardinality. A meta-rule has been defined to let the Contains relation exist between the Room model type and the Workstation model type. As Contains is a one-to-many relation, a single room can contain many workstations, but a single workstation can only be in one room.

The Connects_to relation is an example of a many-to-many relation. A meta-rule has been defined to let the Connects_to relation exist between the switch and router model types. A single switch is connected to many different things, one of them the router. Likewise, a router is connected to many things, one of them being a switch.

The cardinality of relations lets CA Spectrum models be logically linked, associated, or combined in ways that can truly represent real-world computing infrastructures.

Model Type Hierarchy

The model types are built in a hierarchical fashion. The more general model types are built first, and the more specific model types are derived from the general types. A model type is derived using the principal of inheritance. Multiple inheritance is used to derive a model type from multiple base model types.

A derived model type inherits both the attributes and the intelligence of the model type or model types from which it is derived. The derived model type also participates in the same meta-rules as the base model types. In addition, the derived model type uses the same inference handlers that the base model types use.

Model types that are derived from multiple base model types inherit attributes and inference handlers from the base model types using a specific order. As a result, the derived model type cannot inherit an attribute or an inference handler multiple times. Inheritance also determines the initial value of an attribute. The new attributes, both internal and external, and new inference handlers can be added to the derived model type. The derived model type is a more specific type than its base.

Models

Along with storing model types, the knowledge base stores all the models that have been instantiated to represent elements of the computing infrastructure. A model is created by instantiating a specific model type. A copy of the template (the model type) is made, and the copy is then used to represent a real-world element in the computing infrastructure.

When a model is instantiated, the attributes of that model type take on values. The knowledge base also stores the current value for each model attribute. Some of these model attributes are "shared," being common to all elements of the same type, and describe aspects or behavior of the model type. Each model of a given model type has the same value for these shared attributes. The unshared attributes have values which can differ for each model according to the current working condition in the infrastructure. The values of the attributes describe the unique aspects, characteristics, and behavior of the single model.

Associations

When CA Spectrum creates a representation of computing infrastructure components using models, these models do not exist as isolated elements. Models relate to one another as the elements relate to one another in the computing infrastructure. When CA Spectrum instantiates a model, the applicable relations between the model and other models are also instantiated. An instantiated relation is known as an association. The association must follow the meta-rules that define the relation. OneClick and other client applications enforce meta-rules, not the SpectroSERVER.

For example, consider the relationship between the buyer and seller of a house. The nouns (model types) are BUYER, SELLER, and HOUSE. The verbs BUY and SELL capture the potential relationships that can exist between the nouns. These verbs are the relations. The relations need meta-rules to provide meaning. The two meta-rules are BUYER BUYS HOUSE and SELLER SELLS HOUSE. Given these meta-rules, the BUYS, and SELLS relation begin to have some meaningful value to the modeling system.

An association applies a meta-rule to existing models. Assume that BuyerSmith is a model of type BUYER, and SellerJones is a model of type SELLER. NiceNewHouse is a model of type HOUSE. Given the meta-rules, it is valid to set up an association that states BuyerSmith BUYS NiceNewHouse. Likewise, you can set up an association that says SellerJones SELLS NiceNewHouse. Meta-rules, however, do not allow a relationship such as NiceNewHome Buys BuyerSmith, therefore this association is invalid.

More information:

Managed Elements (see page 22)

Inference Handlers

The inference handlers define the behavior and intelligence of a model type. Each inference handler can perform a specific task. A task can be as simple as changing the value of an attribute. Or, a task can also be as complex as discovering all the managed elements on a network segment. An inference handler can also perform a generic task like calculating an average. Or, an inference handler can perform a detailed task specific to a model type, such creating models of LAN switch ports. Basically, the inference handlers are the many pieces of intelligence that are the heart of CA Spectrum. CA Spectrum can offer its many infrastructure management capabilities because of inference handlers.

An inference handler is a C++ code segment that is associated with a model type. An inference handler is typically a dormant piece of code that supports a wide variety of triggers. Once triggered, an inference handler performs a task. The result of the task can be a new piece of data, an altered modeling scheme, or another CA Spectrum subsystem (such as another inference handler) triggered. Once the processing of inference handler ends, it goes into an idle state and awaits another trigger.

The inference handlers specify the behavior for models according to their model type and how the model type reacts to certain conditions. They can define:

- The behavior of a model when it is created, destroyed, or activated.

  Note: A model is activated when it establishes the necessary communication with the managed element that it is modeling.

- The behavior of a model if the values of its attributes change, or if an event is generated on it.
- The behavior of a model when it forms a new association with another model, or when the model is removed from an existing association.
- How to handle certain actions.

The inference handlers are related to model types within the knowledge base, and they execute for instantiated models of that model type. When the external condition of two models of a model type changes in a similar way, the reaction of both models is similar. However, the values of the attributes of a specific model that reflect the status of that model have an impact on the inference handler. The attribute values of one model can be different from the attribute values of the other model. Therefore, even though the external condition is the same and the inference handlers react similarly for each model, the result of the reaction by the two models of the same model type can be different.

For example, an inference handler that is associated with a model type that represents a router is designed to perform one specific task: To create models that represent the interfaces of the router whenever a new router model is instantiated. Once this task ends, the job of the inference handler finishes. The inference handler then waits for the creation of the next router model from this model type, so it can perform this job again.

The router model type has attributes that record the number and type of interfaces that exist on the router. Each instantiated router model represents a specific router in the computing infrastructure. And, it is likely that each router model has different values for these attributes. The number and type of interface models that the inference handler creates, are based on these values. Thus, if multiple router models are created in the knowledge base representing different routers of the type in the network, the same inference handler creates a different (but appropriate) number and type of interface models for each new router model.

The previously mentioned inference handler can once again be triggered when the router model receives a notification that the real-world router has been reconfigured. The reconfiguration can cause the number or type of interfaces on the router to change. When this change occurs, the inference handler recreates these interface models using the new information. This dynamic adaptive modeling capability is an example of one of the fundamental uses of inference handlers throughout CA Spectrum.

More information:

SPEC--Alerts, Events, and Alarms (see page 25)

Actions

CA Spectrum defines a set of operations that can be performed on a model, such as reading or writing an attribute. To expand possible operations, CA Spectrum allows *actions*. An action is any operation that is not part of the basic set of operations that CA Spectrum defines for use with a model. Sending an action to a model causes the model type to react in some way. For example, it can return requested data to the action sender, or it can cause the model type to perform a specific task.

Reporting Database

CA Spectrum Report Manager uses a MySQL database to store data that is used in its reports. Consider the following points about the Reporting database:

- If you install OneClick without installing Report Manager, a local MySQL database server is installed. This local database provides the Distributed Data Manager (DDM) database, the MIB Tools database, the eHealth integration database, and the Reporting database.

  For this installation, the Reporting database is used only to store Service Manager data. For more information, see CA Spectrum Databases (see page 11).

- If you include Report Manager with the OneClick installation, the Reporting database stores the Service Manager data. The database also stores the asset data and historical event and alarm data for reporting purposes.

  Note: The asset data and historical event and alarm data must be stored in the MySQL instance that is on the OneClick web server. The data cannot be stored on a separate computer.

Note: For more information, see the *Report Manager Installation and Administration Guide* and the *Report Manager User Guide*.

The SpectroSERVER and Threads

The SpectroSERVER handles requests from many client applications while simultaneously accessing the disk and the network. For better efficiency, the SpectroSERVER operates using a multithreaded architecture that has less overhead than running separate processes.

The SpectroSERVER creates some threads at startup that terminate only when the SpectroSERVER terminates. The SpectroSERVER creates other threads dynamically and terminates them when they are no longer needed. For example, each time a client connects to the SpectroSERVER or makes a request through an API, a new thread is started.

Typically, it is not necessary to be concerned about this internal threading mechanism. However, this concept can become important on a heavily loaded system when advanced tuning is required to maximize your system throughput.

Managed Elements

The SpectroSERVER uses models to represent managed elements, and these models are based on the model types that are defined in the modeling catalog. Some model types can be instantiated to represent a device, an application, or a host that operates in the computing infrastructure. The SpectroSERVER can communicate directly with these managed elements using SNMP, if appropriate. Some model types are instantiated into models that act as containers and are used to group other models. For example, you can create a LAN model to group certain managed elements on a network segment. Or you can create a Room model to group the managed elements in one room.

A container model can contain either other container models or models that represent managed elements, or both, depending on the container model type. For example, an IPClassB container can contain a model representing a router, and it can also contain several LAN models representing a range of subnets. However, a Building model can only contain container models: a Floor, a Section, or a Room.

The SpectroSERVER uses *management modules* to manage the specific elements of a computing infrastructure. A management module is made up of model types, relations, inference handlers, and support files. A management module uses a series of models to represent each component of the specific type of managed element. The managed element can be represented with a device model. The device functionality is supported with a combination of other types of models, such as application models. Each major functional component of a managed element can be modeled as a separate application or can be incorporated into the device model. An application often corresponds to the functionality of a MIB, or a section of that MIB.

All the models that are used to represent a managed element are based on model types that are defined in the modeling catalog. The associations that the models have with one another are based on the relations and meta-rules that are defined in the modeling catalog. The SpectroSERVER can implement various associations, such as the following examples:

- A container model can contain models representing managed elements or container models
- Connections between device or port models can be established representing physical or logical connectivity
- Application models that support a device model express relations showing what functionality a device provides

Note: Not all model types that are defined in the CA Spectrum knowledge base can be used to create a model in OneClick. Some function only as base model types from which other model types are derived.

Device Discovery

CA Spectrum supports an automatic method of discovering and modeling managed elements and connections within the computing infrastructure. *Discovery* is a OneClick feature that automates the process of discovering and modeling the entities in your IT infrastructure. You can create and edit Discovery and modeling configurations to customize and simplify the process. DIscovery also lets you filter and export the results of Discovery or modeling sessions.

Discovery comprises two components. The Discovery application scans IP address ranges or lists and reads a select group of key MIB objects from each SNMP-enabled managed element encountered. The result is presented to you or is sent to the SpectroSERVER for modeling. The server-side, or back-end, Discovery process uses the key MIB objects of each managed element to determine the best model type to use. Then Discovery creates an instance of that model type to represent that managed element.

After the models are created and activated for the managed elements that Discovery finds, the back-end Discovery process determines their placement and connectivity. Model placement and connectivity are based partially on user-specified options. Based on their IP address, models of bridges and workstations are placed inside a LAN container. These models are then connected to other bridges using spanning tree and source address tables, which are read from the MIBs of the managed elements. Models of routers are placed in network containers or in the Universe. These models are connected to LAN models or to other routers using IP addresses and masks, IP route table information, or proprietary discovery protocol MIBs.

CA Spectrum also lets you create specific models from the OneClick Console. Two methods can be used. The first method uses the IP address or the DNS name of the managed element. With this information, the SpectroSERVER contacts the managed element and retrieves the name, vendor, description, location, and sysOID of the managed element. CA Spectrum then creates a model using the model type that best represents the functionality of this device.

You can also create a model by first selecting a model type as a base. In this case, you provide an IP address or a DNS name so the SpectroSERVER can communicate with the managed element. However, the model type that you select is instantiated, regardless of the SpectroSERVER assessment of managed element functionality. The SpectroSERVER creates all of the appropriate supporting model types and matches the functionality that the MIBs for the managed element describe.

Device Communication Manager

The Device Communication Manager (DCM) is the interface between the SpectroSERVER and the managed elements. The DCM includes various protocol interfaces that communicate with managed elements using a specific protocol. One interface exists for each of the two supported protocols, SNMP and ICMP. When the SpectroSERVER communicates with a managed element, the request is sent to the appropriate protocol interface in the DCM. The DCM, in turn, passes the request to the managed element.

Polling

The SpectroSERVER constantly updates its knowledge of network conditions using polling and logging services. The DCM handles communication with the managed element being polled. Model type attributes are defined as either external (to be obtained from the managed element) or internal (stored either in memory or the database). Some external attributes are defined as *polled*, which means that the SpectroSERVER regularly polls the managed elements. The polling frequency depends polling_interval attribute value that is defined for the model. Values of external attributes that do not have polling set are obtained from the managed element upon each client application or inference handler request.

Note: Polling affects the performance of the SpectroSERVER and the network. Shorter polling intervals can limit the responsiveness of the SpectroSERVER and can generate an unacceptable amount of network traffic.

Logging

Attributes can also be defined as being logged, meaning that their values are written through the Archive Manager to the DDM database. The frequency with which values are logged is based on both the polling_interval and the Poll_Log_Ratio defined for the model.

For example, you can set the Poll_Log_Ratio to 10 and polling_interval to 60. This setting logs the attribute value to the statistics file every tenth poll, or every 600 seconds.

Note: Logging affects the performance of the SpectroSERVER and the network. Smaller logging ratios can limit the responsiveness of the SpectroSERVER and can generate an unacceptable amount of network traffic.

SPEC--Alerts, Events, and Alarms

CA Spectrum is a services and infrastructure management system that notifies you when a fault occurs in a managed element in your network. One way that CA Spectrum accomplishes this functionality is by receiving alerts (usually SNMP traps) from problem areas in the computing infrastructure. CA Spectrum then converts those alerts into events and alarms to be displayed in CA Spectrum applications.

CA Spectrum uses a series of support files that are named event configuration files to indicate how alerts, events, and alarms are processed. CA Spectrum receives alerts from problem areas within the managed infrastructure. CA Spectrum converts alerts into events and alarms, which are displayed in OneClick event and alarm views. Alerts, events, and alarms let CA Spectrum notify you about significant occurrences in your IT infrastructure.

More information:

Inference Handlers (see page 19)

SPEC--Alerts

An alert is an unsolicited message that a managed element sends to CA Spectrum. The primary management protocol that CA Spectrum uses to communicate with managed elements is SNMP. An SNMP-compliant managed element can send an alert, which is known as a trap. Managed elements with SNMP traps enabled can be configured to direct their traps to the SpectroSERVER. The SpectroSERVER uses the source IP address of the trap to identify the model that is associated with that managed element. Once the model is known, the trap is processed as directed by the AlertMap file that is associated with that model type. An AlertMap file exists for most device model types within CA Spectrum. The AlertMap is an ASCII file that is used to map SNMP traps into CA Spectrum events.

SPEC--Events

An event is an object representing an instantaneous occurrence within CA Spectrum. Events usually indicate that something significant has occurred regarding the model or another component. Most device model types have an associated EventDisp event configuration file. An EventDisp file is an ASCII file that indicates how an event is processed. After an AlertMap file converts an SNMP trap into an event, the EventDisp file tells CA Spectrum how to handle this event for this model. The processing of an event can include logging the event and generating an alarm.

SPEC--Alarms

An alarm is an object which indicates that a user-actionable, abnormal condition exists in the managed environment. Usually an alarm is generated as a result of a received trap— that is, when an event has occurred and the EventDisp file specifies alarm generation. A configured watch can also generate an alarm, as can CA Spectrum detecting an abnormal situation that is not based on an event. After the abnormal condition that caused the alarm ends, another event clears the corresponding alarm or you can clear it. Alarm notifications can be sent to applications and inference handlers that need this information. CA Spectrum can examine myriad network events, analyze them, and produce few important alarms.

Landscapes and the Distributed SpectroSERVER

A landscape is the CA Spectrum term for a network domain that a single SpectroSERVER manages. A landscape is composed of the models, associations, attribute values, alarms, events, and statistics belonging to a specific SpectroSERVER. Each landscape that is contained in a network is unique, and a unique landscape handle (ID) identifies each landscape. A landscape icon can represent each landscape in the OneClick Console. The landscape icon provides a graphical representation of a SpectroSERVER knowledge base.

The Distributed SpectroSERVER (DSS) is a powerful modeling feature that enables the distribution of management for portions of a large-scale network. The work can be distributed either geographically or across multiple servers in a single physical location. DSS can improve CA Spectrum performance when managing a computing infrastructure by distributing the network load that management traffic introduces. DSS can also delegate management functions to remote workstations.

Using a DSS, you can create a unified representation of the computing infrastructure, which is composed of multiple landscapes, each with its own local SpectroSERVER. In a DSS environment, a SpectroSERVER client, such as the OneClick Console, can access information from more than one SpectroSERVER simultaneously.

Note: The *Distributed SpectroSERVER Administrator Guide* contains tips on how to segment your network into landscapes.

When you model multiple landscapes using DSS, the database for each landscape must contain identical modeling catalogs. All model types that exist in the modeling catalog for one landscape must also exist in the modeling catalog of every other landscape. Hence, if you install add-on applications in one landscape, the same applications must be installed on every landscape. For example, if you install VPN Manager in one landscape, install VPN Manager in all landscapes.

Administration of all the modeling catalogs in a distributed environment is made easier with the concept of a master catalog. The master catalog is the SpectroSERVER that you designate to update the other SpectroSERVERs in the landscape map. When a change is necessary, it is made to the master catalog. The entire master catalog is manually copied to all other SpectroSERVERs in the landscape map, propagating all changes and keeping the modeling catalogs consistent.

Note: See the *Distributed SpectroSERVER Administrator Guide* for more information.

Chapter 3: Client Applications

This section contains the following topics:

Client Applications Overview

The main CA Spectrum client application is OneClick. A few other CA Spectrum client applications also let you interact with the information that is stored and processed on the SpectroSERVER.

You can use the following client applications when customizing CA Spectrum or integrating with CA Spectrum:

- **AlarmNotifier:** This application is used to forward alarm data to user-defined scripts or third-party applications. For more information, see About AlarmNotifier (see page 31).
- **SANM:** This application is used with the AlarmNotifier to specify policies that filter alarm data sent to user-defined scripts or third-party applications. For more information, see About SANM (see page 33).
- **Report Manager:** This application is a full-featured system that creates reports from data that is extracted from the knowledge base on the SpectroSERVER. Report Manager relies on two more client applications: CA Business Intelligence (see page 34) and InfoView (see page 35).

Each of these applications also has its own *User Guide*, which is available on the CA Spectrum documentation bookshelf.

The following applications are not CA Spectrum clients, but they are required for some optional CA Spectrum customization and integration.

- **Process daemon:** The process daemon is a process launching and tracking daemon that lets CA Spectrum control various processes running on a workstation. The process daemon starts processes when requested by an application, such as the Control Panel. The process daemon can also start processes on system boot when configured to do so. The process daemon automatically restarts critical processes when they stop unexpectedly. The CA Spectrum Control Panel is the only executable actively launched by you, the CA Spectrum user. The process daemon launches all other applications following a request by the user or another application. The process daemon operates in the background and is transparent to you. The process daemon automatically starts during CA Spectrum installation and whenever the system is started.
- **Model Type Editor:** This application is used to derive new model types that support the development of new management modules.

OneClick Console

The OneClick Console displays information from the SpectroSERVER using icons and views. Icons are illustrations of the models that are defined to represent the managed elements of your computing infrastructure. Views are the various ways in which data from the SpectroSERVER is organized for display.

OneClick Console Icons

Icons are graphic representations of instantiated models that are based on model types from the CA Spectrum modeling catalog. The various icons can represent individual managed elements, groups of managed elements, geographic locations, users, landscapes, connections between models, and so on. Pipes are a special type of icon and are used to represent the connectivity between managed elements.

General information about a model, such as the model name and model type name, is visible on the icon. Detailed information about a model is found within various icon subviews that are accessed by double-clicking the icon. Some icons use color to indicate the condition of the managed elements that they represent.

Hierarchical Views

A view in CA Spectrum is a way to organize data so it can be displayed or manipulated. Hierarchical views represent ways to structure your network data. When structuring your network data in the XML file, you select from elements that represent each of the hierarchical views. Two types of hierarchical views exist: Topology and Location.

Topology View

The Topology view is really an abstraction of networking components. When working with this view, you represent the physical or logical components of your network and group these components, while factoring in their logical connectivity. You can also represent connections graphically, using pipes that show how devices are connected at the port or device level. In the OneClick Console, this view appears as the Universe topology.

Note: For more information about the Universe topology view, see the *Modeling and Managing Your IT Infrastructure Administrator Guide*.

Location View

The Location view organizes your network data by physical location. Using this view that you can depict your network in geographical terms. You can start with your global offices and can go right to the wiring closet on each floor of each building in each region where your offices are located. In the OneClick Console, this view appears as the World topology.

Note: For more information about the World topology view, see the *Modeling and Managing Your IT Infrastructure Administrator Guide*.

About AlarmNotifier

AlarmNotifier is a SpectroSERVER-client application that installs with core CA Spectrum components. The AlarmNotifier application connects to a single SpectroSERVER and invokes scripts that provide notifications about CA Spectrum alarm status.

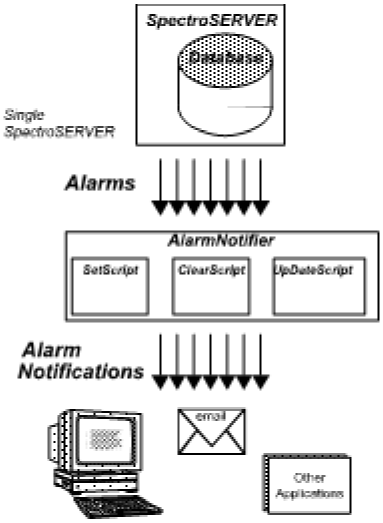
Start AlarmNotifier from a terminal shell command prompt. Once started, it continuously displays output from scripts that are invoked whenever alarms are either set, cleared, or updated. AlarmNotifier provides the following features for CA Spectrum:

- Single SpectroSERVER alarm monitoring.
- Three scripts that generate alarm information: SetScript, ClearScript, and UpdateScript.

  These scripts contain settings that can be customized for your environment.
- Resource file parameters that can be configured to modify AlarmNotifier operational features.

Alarm Monitoring Process

AlarmNotifier supplements CA Spectrum alarm monitoring and notification features. The following diagram illustrates the relationship between AlarmNotifier and CA Spectrum:



CA Spectrum performs some alarm functions, while AlarmNotifier performs others. CA Spectrum polls the modeled network elements and updates the status information about each element that is stored in the SpectroSERVER database.

CA Spectrum generates an alarm when it receives a trap from the network or when it detects a critical status change in a network-element model. In the OneClick Topology view, the condition of the model icon changes from green to another color to indicate alarm severity. CA Spectrum posts information about the alarm in the Alarms tab. Event information for the alarm appears in the Events tab in the OneClick Contents panel.

When AlarmNotifier is started, it registers with CA Spectrum. Then a model named AlarmNotifier of type ClientApp is created. This model is not visible in any of the CA Spectrum Topology views. However, you can see it in the Events tab. The Events tab displays information such as the application start and stop time for this model.

AlarmNotifier queries the SpectroSERVER and requests information about existing alarms. AlarmNotifier runs scripts and generates notifications about existing alarms.

Each time an alarm is set, cleared, or updated, AlarmNotifier receives information from the SpectroSERVER and invokes the relevant script. AlarmNotifier scripts can initiate email notifications of alarms that are sent to network personnel. They can also transmit alarm information to third-party applications.

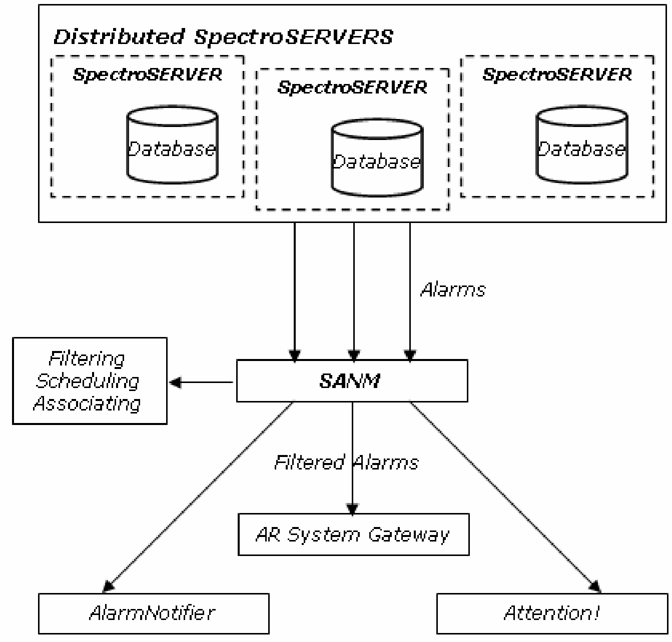
About SANM

The Alarm Notification Manager (*SANM*) is a CA Spectrum component that enhances the functionality of CA Spectrum alarm-processing applications. Multiple alarm-processing applications are available for CA Spectrum, including AlarmNotifier and Attention!. These applications respond to CA Spectrum alarms by sending email notifications, creating trouble tickets, and more. SANM lets you create and associate alarm notification policies with applications.

How CA Spectrum Monitors Alarms

CA Spectrum, alarm-processing applications, and SANM work together in the alarm monitoring process.

The following diagram shows the alarm monitoring process:



The following workflow describes how CA Spectrum monitors alarms:

1. CA Spectrum polls the modeled network elements and updates the status of each element in the SpectroSERVER database.
2. CA Spectrum generates an alarm when it receives a trap from the network, or when it detects a critical status change in a network model. In the OneClick Console, the model icon changes from green to another color that indicates the alarm severity level.
   - CA Spectrum posts specific information for each alarm on the Alarm Details tab of the Component Detail pane.
   - CA Spectrum posts alarm event information to the Events tab of the Component Detail panel.
3. Data about alarms that CA Spectrum has generated is passed to SANM. SANM lets you create and associate alarm notification policies with alarm processing applications. In addition, the SANM Schedule subview lets you schedule application and policy associations and automates the association process.
4. SANM passes the alarm information to alarm processing applications only when the alarm types specified in the policies occur.

Reporting with CA Business Intelligence (CABI)

CA Spectrum Reporting uses CA Business Intelligence (CABI) to display reports.

CABI is a reporting and analytic software package that CA Spectrum and other CA products use to present information. CA Spectrum uses CABI to integrate, analyze, and present information that is required for effective enterprise IT management, through reports.

CABI includes SAP BusinessObjects Enterprise XI, which is a complete suite of information management, reporting, query, and analysis tools.

CABI installs SAP BusinessObjects Enterprise XI as a standalone component. The installation runs independently of CA Spectrum and other CA products, allowing various CA products to share Business Intelligence services. CABI installation is a distinct and separate activity within the overall CA product installation process.

Note: For more information, see the *CA Business Intelligence Implementation Guide* and the *CA Business Intelligence Release Notes*.

InfoView Report Management

BusinessObjects Enterprise InfoView (InfoView) is a web-based interface that lets you manage reports with the following features:

- Browsing and searching capabilities.
- Content access (creating, editing, and viewing).
- Content scheduling and publishing.

The InfoView functions like a Windows application rather than a simple web application. The InfoView toolbar dynamically changes to provide actions through context menus consistent with the function you want to perform. InfoView provides context menus, by simply clicking the right mouse button. You can double-click items in a window to execute default actions. Report structures are consistent and provide powerful security and authorizations.

InfoView provides access to the WebIntelligence (WEBI) designer. This designer lets you create customized reports with a simple drag-and-drop interface. Custom data object selection with effective filtering options, enables powerful reporting capabilities for your environment.

You can access InfoView from the OneClick home page or directly from a Web browser. The typical URL format is as follows:

```
http://<hostname>/InfoViewApp
```

Note: For more information, see the *Report Manager User Guide*.

Appendix A: Attribute and Relation Definitions

This section contains the following topics:

Attributes

The concept of an attribute and its role regarding models and model types is outlined in Model Type Attributes (see page 14).

Many attributes in the knowledge base are counterparts of MIB variables. External attributes directly correspond to specific MIB variables. Internal attribute values can be derived from the values of MIB variables. These values have often undergone some mathematical calculations to arrive at their CA Spectrum value. The naming conventions typically make the relationship between the attribute and the MIB variable evident.

The following tables show the attributes that have been defined to help you integrate other applications with CA Spectrum. These attributes are used when creating a management module, using the Southbound Gateway Toolkit, or using the Modeling Gateway Toolkit. Each table is a quick reference with attributes grouped by functionality.

Note: You can access specific information about attributes and relations from the Model Type Editor. For more information, see the *Model Type Editor User Guide*.

Application Model Discovery

Attribute	Attribute ID	Found On
default_attr	0230006	Application model types

Device Model Discovery

Attribute	Attribute ID	Found On
DeviceNameList	0x1293E	Device model types
DeviceType	0x23000e	Device model types
Disposable_Precedence	0x114e2	Device model types
Enable_IH_Spec_Dev_Name	0x3d0062	Device model types
Enable_IH_Device_Name	0x3d0008	Device model types
Image_Index	0x3d0001	Device model types
System_OID_Verify	0x110bb	Device model types
System_OID_Verify_List	0x12910	Device model types

General Model Type Information

Attribute	Attribute ID	Found On
CompanyName	0x118b8	Device and application model types
Description	0x230017 and 0x118bc	Device, application, and interface model types
DeviceType	0x23000e	Device model types
Manufacturer	0x10032	Device model types
MMName	0x1196a	Device and application model types
MMPartNumber	0x1196b	Device, application, and interface model types
Model_Class	0x11ee8	Device model types
Model_Name	0x1006e	Device, application, and interface model types
Modeltype_Name	0x10000	Device, application, and interface model types
Vendor_Name	0x11570	Device, application, and interface model types

Network Information

Attribute	Attribute ID	Found On
Network_Address	0x12d7f	Device, application, and interface model types
Network_Mask	0x12dbc	Device, application, and interface model types

Polling Information

Attribute	Attribute ID	Found On
polling_interval	0x10071	Device, application, and interface model types
poll_log_ratio	0x10072	Device, application, and interface model types
pollingstatus	0x1154f	Device, application, and interface model types

Port Identification

Attribute	Attribute ID	Found On
ifAlias	0x11f84	Interface model types
if_Index	0x11348	Interface model types
ifName	0x11f60	Interface model types
if_Phys_Addr	0xd0399	Interface model types
ip_address	0x12dbb	Interface model types

SNMP Information

Attribute	Attribute ID	Found On
Community_Name	0x10024	Device, application, and interface model types

Attribute	Attribute ID	Found On
CommunityNameForSNMPSets	0x11a7f	Device, application, and interface model types
isManaged	0x1295d	Device model types
Security_String	0x10009	Device, application, and interface model types

Attribute Descriptions

This section provides more information about the attributes that are outlined in the previous tables.

Note: You can access more information about each attribute by using the Model Type Editor. Details, such as the attribute ID, the data type of the attribute value, or the attribute flags that are set for the attribute, are available. See the *Model Type Editor User Guide*.

Several attribute flags can be set for each attribute including these main ones:

- **External:** When set to TRUE, this flag indicates that the value for this attribute is maintained outside of the SpectroSERVER. This setting also indicates that an update of the attribute value is done either at a user request or at a polling interval.
- **Readable:** When set to TRUE, this flag informs the SpectroSERVER that a client or other application is allowed to read this attribute value from the SpectroSERVER. If the External flag is set, set this flag according to the MIB definition of the Readable variable for this attribute. If the External flag is not set, set this flag as desired.
- **Writable:** When set to TRUE, this flag informs the SpectroSERVER that a client or other application can write this attribute value to the SpectroSERVER database. If the External flag is set, set this flag according to the MIB definition of the variable for this attribute. If the External flag is not set, set this flag as desired.
- **Shared:** When set to TRUE, this flag declares that one value exists for this attribute and that all models of the current model type share it. The value is not duplicated in memory or in the database for each model.

The following list contains descriptions for the attributes that one or more of the CA Spectrum integration points uses or references.

Community_Name

Identifies the SNMP community string that is used when CA Spectrum attempts to communicate with a managed element using SNMP. This attribute is evaluated when performing SNMP gets. If the attribute CommunityNameForSNMPSets is empty, this attribute is used when performing SNMP sets, too.

CommunityNameForSNMPSets

Specifies the community name that is used when performing SNMP sets. If left blank, the Community_Name attribute value is used for SNMP sets.

CompanyName

Specifies the company name the device and application model types use. This value is set equal to the name of the company that developed the model type.

default_attr

Identifies the applications that a particular managed element supports. This attribute is used in the application Discovery process. The value of the default_attr is set equal to the attribute ID of an attribute that represents a MIB object which uniquely identifies the MIB.

Description

Provides a textual description of the model type.

DeviceNameList

Contains the device names that correspond to the OIDs in the SysOIDVerifyList attribute. To use this attribute to set the device name, set the Enable_IH_Device_Name attribute and the Enable_IH_Spec_Dev_Name attribute to TRUE.

DeviceType

Holds the name of the device type when the model type represents only one specific type of device. The value of this attribute is displayed in the field at the bottom of the icon for the model. If multiple device types exist for a given device, use the DeviceNameList attribute.

Disposable_Precedence

Contains a value that the CA Spectrum device discovery mechanism uses to resolve conflicts in the device model type selection process. A conflict occurs when multiple model types have a System_OID_Verify value that matches the SystemObjectID of the managed element. To resolve the conflict, Discovery uses the model with the highest Disposable_Precedence value.

Enable_IH_Spec_Dev_Name

When the Enable_IH_Spec_Dev_Name attribute is TRUE, CA Spectrum uses the Enable_IH_Device_Name inference handler to determine the vendor name using the enterprise number from the device.

Enable_IH_Device_Name

When the Enable_IH_Device_Name attribute is TRUE, CA Spectrum uses the Enable_IH_Spec_Dev_Name inference handler to read the device System Object ID to determine the product name.

ifAlias

Corresponds to this value from the MIB II Interface table.

ifIndex

Corresponds to this value from the MIB II Interface table.

ifName

Corresponds to this value from the MIB II Interface table.

if_Phys_Addr

Corresponds to this value from the MIB II Interface table.

Image_Index

Links a GnSNMPDev model with an icon image that represents the model. Possible values and their corresponding images are as follows:

1

Generic Device

2

Bridge

3

Router

4

Hub

5

PC

6

Terminal Server

7

Workstation

8

Switch

ip_address

Specifies the IP address that is associated with an interface model.

isManaged

Indicates if a device model is managed. When this attribute is set to TRUE, CA Spectrum manages this device using SNMP communication.

Manufacturer

Used with a device model to show the manufacturer responsible for the device.

MMName

Holds the management module name for device and application models.

MMPartNumber

Contains the part number that the management module developer has assigned to the management module. Most device, application, and interface models use this attribute.

Model_Class

The Model_Class defines the type of device that the model represents. The following table lists the model classes that are available in CA Spectrum and their respective integer identifier. For example, the model class SWITCH uses the identifier 2.

0

UNKNOWN

1

OTHER

2

SWITCH

3

ROUTER

4

SWITCH_ROUTER

5

HUB

7

LINK

8

NETWORK

9

WORKSTATION_SERVER

10

CONTAINER

11

CHASSIS

12

PINGABLE

13

MAC

14

SNMP

15

PORT

16

USER

17

APPLICATION

18

COMPONENT

19

LANDSCAPE

20

ROUTER_APP

21

SWITCH_APP

22

BRIDGE_APP

23	MIB_APP
24	RMON_APPL
25	UNIX
26	NT
27	FIREWALL
28	IDS
29	SECURITY_SCANNERS
30	ANTI_VIRUS_APPLICATION
31	PKI_SYSTEMS
32	PACKET_SNIFFER
33	SYSLOGS
34	RESPONSE_TIME_TEST
35	RESPONSE_TIME_TEST_HOST
36	TRANSPORT_SERVICE
37	GENERIC_TL1_DEVICE
38	VOIP

39

CMTS

40

WIRELESS

41

CABLE_MODEM_MTA

42

VPN

43

DSL

44

MULTIPLEXOR

45

SAN

46

PBX

47

USER_CUSTOMIZATION

48

PRINTER

49

TRANSPORT_DEVICE

50

SERVICE_MGT_COMPONENT

51

SLA_COMPONENT

52

CUSTOMER

53

PROCESS

54

DIAGNOSTIC_DATA

55

DIAGNOSTIC_COMPONENT

56

HOST_CONFIGURATION

57

DIAGNOSTIC_SCRIPT

103

POWER_SUPPLY

104

AMPLIFIER

105

LINE_MONITOR

106

TEST_POINT

107

FIBER_NODE

108

HEFIBER

109

IP_PHONE

110

TELCO

111

LOAD_BALANCER

112

LMT

113

FILE_SERVER

114

ENVIRONMENTAL

115

AUTOMATIC_TELLER_MACHINE

Model_Name

Specifies the name that is given to the model. A model name distinguishes the model from others of that type.

Modeltype_Name

Specifies the textual name of the model type.

Network_Address

Contains the network address of the device model. CA Spectrum uses this value to identify and communicate with the model. This value is usually an IP address.

Network_Mask

Further identifies the logical location for a device model on the network.

polling_interval

Used by most device, application, and interface models. The value of this attribute identifies the number of seconds that must pass between polling requests. You can set this attribute in the Model Information view.

poll_log_ratio

Identifies the polling cycle that logs the polled attributes. For example, a value of 10 means that the polled attributes are logged every tenth polling cycle. This attribute is found on most device, application, and interface models.

pollingstatus

Controls if polling occurs for the model. Most device, application, and interface models use this attribute. A value of TRUE means that polling is enabled for the model. A value of FALSE means that polling is disabled for the model.

Security_String

Defines the SNMP community strings that have access to this model. Most application, device, and interface models use this attribute.

System_OID_Verify

Contains a System Object ID value that Discovery uses to determine the appropriate model type for representing a device. CA Spectrum assigns the model type when instantiating a model to represent a managed element. CA Spectrum identifies the appropriate model type by matching the System Object ID of the managed element to the System_OID_Verify value for that model type. If multiple model types are found to have a matching value, the model with the highest Disposable_Precedence value is used. If a model type is used to represent a family of devices, the SysOIDVerifyList attribute is used instead of the System_OID_Verify attribute.

SysOIDVerifyList

Contains a list of System Object ID values that Discovery uses to determine the appropriate model type for representing a device. CA Spectrum assigns the model type when instantiating a model to represent a managed element. CA Spectrum identifies the appropriate model type by matching System Object ID of the managed element to a value from the model type SysOIDVerifyList list. If multiple model types are found to have a matching value, the model with the highest Disposable_Precedence value is used.

Vendor_Name

Identifies the vendor of the managed element. Most device, application, and interface models use this attribute.

Relation Descriptions

The following relations are among the most important for the CA Spectrum functionality:

Connects_to

The Connects_to relation is a many-to-many relation type. Connects_to forms a connection between two models. This relation establishes a connection between a port model and a device or topology model.

Contains

The Contains relation is a one-to-many relation type. This relation lets a model contain other models. The Location models use the Contains relation to determine the location or device models that can be contained within the location model. CA Spectrum tests the rules for this relation when you attempt to add or copy models to the Location models.

HASPART

The HASPART relation is a many-to-many relation type. HASPART establishes an association between a device model and the models that represent the components of the device. Typically, these components are the interface models of the device. However, this relation can also establish an association between a component model and its components.

Links_with

The Links_with relation is one-to-many relation type. This relation is used to represent a resolved connection between two models. This relation is found typically between two-port models. Both the Link view and Live Pipes rely heavily on this relation.

Manages

The Manages relation is a one-to-many relation type. This relation forms an association between an application model and the device model that is running it. The Manages relation can also form an association between an element management system model and the element models of this system.

Provides

The Provides relation is a one-to-many relation type. This relation identifies which applications provide which subapplications. The Provides relation can also be used to determine which application provided a particular subapplication.

More information:

Relations (see page 15)

Glossary

action

An *action* is any operation that is not part of the basic set of operations that CA Spectrum defines for use with a model.

alarm

An *alarm* is an indication that an abnormal condition exists for a model.

alarm severity

An *alarm severity* is a value found in an EventDisp file that indicates the condition of a model. Conditions represent the presence and seriousness of an alarm. The valid values are 0 through 6 and all represent a different colored condition.

alert

An *alert* is an unsolicited message that is sent from a managed element to the SpectroSERVER.

alert code

An *alert code* is a string that identifies an alert. An alert that is received from an SNMP source contains an alert code that consists of a generic trap followed by a dot (.) followed by an enterprise-specific trap.

AlertMap file

An *AlertMap file* exists for each model type. This file maps incoming SNMP trap data to CA Spectrum events.

application programming interface (API)

An *application programming interface (API)* is a set of routines that are used to make calls to another software package.

association

An *association* is a link that is formed between two models by a relation.

asynchronous call

An *asynchronous call* is a call to a method that begins (but does not necessarily complete) a requested operation before allowing a program to continue. At some point, the requested operation completes and notifies the program. In the meantime, the program and the requested operation can both proceed simultaneously.

attribute

An *attribute* is the declarative knowledge in CA Spectrum that defines a model type. Attributes are defined using the Model Type Editor.

CA Spectrum Extensions Integration (SEI) Toolkit

The *SEI Toolkit* is a set of tools for packaging and distributing an integration so that it can be installed on other CA Spectrum host machines.

client

A *client* is the application process in a client-server architecture.

client-server architecture

Client-server architecture is a system design that is based on the relationship between a service-providing process (the *server*) and a service-using process (the *client*).

CORBA (Common Object Request Broker Architecture)

CORBA is a software component architecture that lets various and different software components communicate. The software components can be written in different languages (C, C++, Java, Smalltalk, COBOL, ADA, Lisp, Perl, Tcl, Eiffel, Python). The software components can also reside on different machines and can be written for different operating systems.

custom installation script

A *custom installation script* is a script that can be included in a VCD. The script executes as the integration package is being installed on the SpectroSERVER.

database

A *database* is a collection of interrelated data that is organized to facilitate efficient and accurate inquiries and updates.

database management system

A *database management system* is a software package that organizes and maintains a database.

developer ID

A *developer ID* consists of a 14-character developer name and a four-character prefix. The developer name must be alpha-numeric and can include underscores, however, no other punctuation marks are allowed. The prefix must also be alpha-numeric, however, no underscores or other punctuation marks are allowed. CA Spectrum uses *developer IDs* to verify that the objects, such as model types, attributes, or relations, created by users or integrators have unique identifiers. These objects can therefore be distributed to other users without conflict.

To obtain a developer ID, contact CA Support.

Development API

The *Development API* is the CORBA-based application program interface (API) to the SpectroSERVER. The Development API is also referred to as the SpectroSERVER Object Request Broker (SSORB) interface because it depends on an object request broker (ORB).

device

A *device* is a managed element.

Device Communications Manager (DCM)

The *Device Communications Manager (DCM)* is a multiprotocol communication engine in the SpectroSERVER that handles the communication with all managed elements, regardless of their protocol. The DCM translates the SpectroSERVER requests into protocols that the individual devices understand.

Discovery

Discovery is a OneClick feature that automates the process of discovering and modeling the entities in your IT infrastructure. You can create and edit Discovery and modeling configurations to customize and simplify the process. DIscovery also lets you filter and export the results of Discovery or modeling sessions.

Distributed SpectroSERVER (DSS)

The *Distributed SpectroSERVER (DSS)* is a modeling feature that uses the concept of landscapes to improve CA Spectrum performance when managing a large computing infrastructure. DSS technology distributes the load that management traffic introduces and lets you delegate management functions to remote workstations.

Edit Mode

Edit Mode lets you edit the current view in CA Spectrum. Selecting Edit Mode displays the File and Edit options in the menu bar.

element management system

An *element management system* lets you provision, manage, and monitor elements in a network infrastructure.

event

An *event* is a significant message from the SpectroSERVER.

event code

An *event code* is a hexadecimal number that uniquely identifies an event.

event data template

An *event data template* is a series of integers that Southbound Gateway uses to format variable data coming from an alert.

event format file

An *event format file* contains the message about the event that appears in the Events tab in the OneClick Console when the event occurs.

event severity

Event severity is a numeric value that is found in an EventDisp file entry that indicates the seriousness of an event. Valid values are from 0 through 100, with 0 being the least severe.

event variable ID

An *event variable ID* is an ID that represents the event variable data. Use of this ID returns the event variable value.

EventAdmin

EventAdmin is the Southbound Gateway model type that represents a third-party management system.

EventDisp file

An *EventDisp file* exists for each model type and determines how CA Spectrum processes events for that model type.

EventModel

An *EventModel* is a model type in CA Spectrum that represents a unique alert source within a Southbound Gateway integration.

ICMP (Internet Control Message Protocol)

ICMP supports packets that contain error, control, and informational messages.

icon

An *icon* is a visual representation of something that is displayed on a screen.

index file

An *index file* is a file that is created using the CA Spectrum Extension Integration (SEI) Toolkit and defines the components of the integration. The index file indicates where the integration components exist on the host and their location on the customer host.

Inductive Modeling Technology (IMT)

Inductive Modeling Technology (IMT) is a CA set of artificial intelligence techniques. These techniques allow a computing infrastructure of arbitrary complexity to be modeled such that every element is given intelligence.

inference handler

Inference handlers are the intelligence behind CA Spectrum. Inference handlers monitor the changes in the environment, the changes between related models, and the changes in relationships between models and attributes.

instance variable

The *instance variable* stores the instance portion of the OID. If the variable binding identifies a particular object from a table variable within the trap MIB, it likely includes an instance ID.

instance variable ID

An *instance variable ID* is the integer that identifies the instance variable in the OID map. Use of this ID returns the instance variable.

instantiate

In object-oriented design, to *instantiate* is to create a particular occurrence of something.

intelligence circuit

An *intelligence circuit* is a collection of inference handlers that defines the behavior of a model type.

knowledge base

The *knowledge base* is the collection of everything that CA Spectrum can model and managed including concepts, relationships, declarative knowledge, and procedural knowledge.

knowledge base management system

The *knowledge base management system* is the software that is used to define and manage the information in the knowledge base.

landscape

A *landscape* is all the data that is specific to any one virtual network machine (VNM) in a single network. The term also identifies the network domain that is managed by a single SpectroSERVER. In OneClick, a landscape is the network view of one SpectroSERVER.

managed element

A *managed element* is an item or device whose status is being monitored and controlled. A managed element can be a network device, host system, application, service, or other computing infrastructure component.

managed object

A *managed object* is a variable on a managed element containing one piece of information about the node. Each node can have several objects.

management agent

A *management agent* is an implementation of a management protocol that exchanges information for the managed element with the management station.

Management Station Access Provider (MSAP)

Management Station Access Provider (MSAP) is a software task that provides an object with access to a management station.

meta-rules

Meta-rules specify which model types can participate in a relation.

MIB (Management Information Base)

A *MIB (Management Information Base)* is a database that resides on a network device and represents that device as a hierarchical collection of objects. A MIB object represents an individual element of information, such as the uptime of a device. MIBs themselves are text files with special syntax.

mkcd

The *mkcd* is a tool that is a part of the CA Spectrum Extension Integration (SEI) Toolkit. This tool finishes the VCD by adding a version number and making the VCD installable on a CA Spectrum host.

mkmm

The *mkmm* is a tool that is a part of the CA Spectrum Extension Integration (SEI) Toolkit. This tool creates the VCD using the index file and the applicable component files.

model

A *model* is a collection of information that forms a specific occurrence of some basic defined type. In CA Spectrum, models are instances of model types.

model type

A *model type* is a template that describes the attributes, actions, and associations that are related to a managed element in CA Spectrum.

Model Type Editor (MTE)

The *Model Type Editor (MTE)* is the primary tool that defines the concepts, relationships, meta-rules, and declarative knowledge in the CA Spectrum knowledge base.

Modular Object Oriented Threads (MOOT)

Modular Object Oriented Threads (MOOT) is a task control manager that is used in the SpectroSERVER.

Navigation Mode

Navigation Mode is the moving from one view to the next in CA Spectrum. The menu bar contains the File and View options when in Navigation Mode. Navigation Mode does not allow editing of a view.

network management agent

A *network management agent* is an implementation of a management protocol that exchanges information about the managed element with the management station.

network management protocol

A *network management protocol* is the means by which the management station and the managed elements exchange information.

network management station

A *network management station* is the host system or workstation that is running the network management applications and protocol.

object-oriented design (OOD)

Object-oriented design (OOD) is design that encompasses the process of breaking a system into parts. Each part represents some class or object from the problem domain. OOD is applied by viewing the world as a collection of objects that cooperate with one another to achieve some desired functionality. OOD typically includes a notation for depicting logical or physical and static or dynamic models of the system under design.

OID

An *OID* is an identifier for a managed object.

OID map

An *OID map* is the syntax that is used to map an alert variable to an event variable.

pipe

A *pipe* is an icon that represents a connection between two devices or ports.

pollable attribute

A *pollable attribute* is an attribute for which the VNM regularly queries the managed element to obtain current values. Attributes are defined as pollable or non-pollable through the Model Type Editor.

probable cause file

The *probable cause file* is an ASCII text file that defines the symptoms, probable causes, and recommended corrective actions for an alarm. When an event generates an alarm, the text in the associated probable cause file is displayed on the Alarm Details tab in OneClick.

procedural knowledge

In CA Spectrum, *procedural knowledge* is information that defines how a concept behaves or reacts to environmental changes.

protocol

A *protocol* is a set of rules that computers use to communicate with each other.

relation

A *relation* is information describing the connection that models have with each other.

SANM

The Alarm Notification Manager (*SANM*) is a CA Spectrum component that enhances the functionality of CA Spectrum alarm-processing applications.

server

A *server* is a process that provides services in response to a client request in a client-server architecture.

SNMP (Simple Network Management Protocol)

SNMP (Simple Network Management Protocol) is a network protocol that is used to monitor devices and applications on a network.

SNMP trap

An *SNMP trap* is an occurrence that is either broadcast or directed to a network management application, notifying the application of device or network activity. SNMP-enabled devices or applications generate traps.

SpectroSERVER

The *SpectroSERVER* is CA Spectrum software that controls the communication with the database and the managed elements. The SpectroSERVER also acts as an interface between applications (such as the user interface) and information that the database and devices provide. The major components within the SpectroSERVER include the Device Communications Manager (DCM) and the Virtual Network Machine (VNM).

SSORB client

The *SSORB client* is an application that accesses the CA Spectrum CORBA interface, SSORB.

synchronous call

A *synchronous call* is a call to a method that performs the entire requested action before a program can continue running. The program continues only after the completion of the called method.

timing interval

A *timing interval* is the frequency with which the current view updates displayed model attribute information. The default is one request per model every 5 seconds for some number of attributes.

value variable ID

A *value variable ID* is an ID that is used to retrieve the value of a variable binding that is sent with an SNMP trap.

variable bindings

A *variable binding* is variable data that is sent as a part of an SNMP trap.

VCD (virtual CD)

A *VCD (virtual CD)* is a series of files that the CA Spectrum Extension Integration (SEI) Toolkit creates. The VCD lets developers easily package and distribute their integrations.

view

A *view* is one of many representations of the network landscape.

virtual network machine (VNM)

Within the SpectroSERVER, the virtual network machine (VNM) is the software level that provides access to data regardless of where the data is stored. The data can be stored in the database, the VNM memory, or any of the managed elements on the network. The VNM also embodies the CA Spectrum intelligence that is known as the Inductive Modeling Technology (IMT).

Index