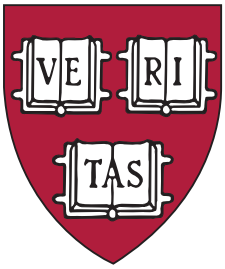
HARVARD
UNIVERSITY



Information Technology

Integer Discovery Service Description

Thursday, July 24, 2014

Table of Contents

Project Integer[1] Discovery Service

Version	Date	Description
0.1	4/21/2014	Initial version.
0.2	4/23/2014	Changed table about characteristics of a Service and a ServiceElement based on input from Loren. Fixed missing information about ServiceElement change and other bugs from Dave T. Finished section on overlapping discovery rules and User Interaction with the Discovery Service. Clarifications to definition of ServiceElement. Added information about InterDeviceLinks and ServiceTechnology. Reorganized and added section on Integer Terms and Concepts of Operation. Added appendices.
1.1	5/7/2014	Added a place holder for relationships - will be updated to a non-UML version.
1.2	5/9/2014	Removed category, no longer needed.
1.3		Reference section update, format edits.
1.4	7/24/2014	Associated Selection Class with InventoryRule, minor change.

Introduction and General Description

In the context of management systems, the term discovery describes functions that traverse a network to find connected systems, details of the topology of that network, and sometimes additional information about the connected elements. In this sense, the Integer system is the same as these other systems. It is also like these other systems in that discovery is often the first function developed in a multi-function system since it is not possible to perform management functions until the system knows which systems are in its environment.

Where the Integer discovery system is different is that the information it collects about each system is more detailed and customizable. These features are present to support the Integer architecture designed for a wide range of integrated functions across a large range of device types and software services. Additionally, the Integer discovery service is designed to collect and integrate information from Layer 1 up through Layer 3 and above. I also integrates the ability to identify information flow

[1] The project, Integer, is an attempt to create a unified whole from the separate protocols, data elements and software systems we use to operate our increasingly complex computing environments. See: http://www.thefreedictionary.com/integer. Also see: http://en.wiktionary.org/wiki/integer#Latin

modification systems beyond standard layer 2 and 3 'routing'. For example the impact of firewall, NAT, and load balancing functions.

The Integer D*iscoveryService*[2], along with the *Topology*, *ServiceElementDiscovery* and *Inventory* managers will be adapted to several environments and types of discovery over time. This document is a high-level summary of the first supported environment, 'standard' IPv4. Subsequent versions of this document will reflect enhancements made for cloud providers such as AWS, extension to storage systems, and various virtualization environments like VMware or OpenStack.

Intended Audience

This document is intended for engineers familiar with the technologies Integer is managing and also have a basic acquaintance with software engineering principles used in its implementation. It is not a user-guide or general requirements document. It is intended to describe general concepts of operation for people that will eventually use Integer and/or be involved in developing it or plug-in modules.

Use of Italics and Reference Documentation

Throughout this document, CamelCase terms are written in *italic* as an indicator that they reference classes found in the Integer system design in UML. Understanding UML or the Java Integer uses to implement it is not necessary to a full understanding of this document. The information is provided for those that wish to explore further. Appendix E is a listing of reference documents that contain additional details about Integer and may be helpful in understanding this document.

Integer Terms and Concepts

This section describes a few key terms and concepts of operation necessary to the understanding of the Integer *DiscoveryService*.

What's a ServiceElement?

In the Integer system, a *ServiceElement* could be anything from a large router to a virtual port or a server that may host many applications. Key characteristics of a *ServiceElement* are:

- *ServiceElements* can exist alone or in a containment hierarchy. Alternatively they may have a referential relationship with other *ServiceElements* in the device. This is the case with some ports in some vendor implementations. Other types of relationships may be defined over time if needed.
- Hierarchical relationships can exist in routers, servers and other systems. They might have a main box that has a chassis, that can have boards, storage systems, etc. Each of these elements may have sub service elements. In the case of a card, it might have several ethernet ports, each of these might have sub-ports which are also *ServiceElements*.
- *ServiceElements* can be physical or virtual.

[2] See the section on "Use of Italics and Reference Documentation" for italic conventions.

- Each service element has information that lets the system know what 'child' elements it has and if it has any 'parent' elements. If there are no parent elements, or referential relationships then it would be what we have historically called the device, like a server or router.
- *ServiceElements* have from 1 to many capabilities/attributes.

There are a number of attributes associated with a *ServiceElement*, some of which are added by operational personnel:

- Location - some of this is filled in by the discovery system using information found in the SysLocation MIB Object.
- Security Level - the classification of information (e.g., HRCI) stored on the device. This would be filled in by the user, or perhaps in the future based on other information (e.g, network, location, etc.).
- State information
- Environment (e.g., production, test) - same as security level, mostly by hand at first.
- Criticality (a backbone core device - high criticality) - same as security level, mostly by hand at first.
- What type of service element it is - *ServiceElementType*. Whenever possible, Integer will attempt to fill in this critical piece of information. In those cases where it can not, the *ServiceElement* will be added and flagged so that Integer can be taught how to distinguish this *ServiceElementType* in the future (see discussion on Incomplete Information and *ServiceElementType* Identification.

What's a ServiceElementType?

From a management software perspective, a system could be defined in terms of the information you can collect from it and the commands or information that you can send to it[3]. In the Integer system, these attributes are called *Capabilities*. Different systems have different ways of accessing [protocols] these attributes, or the way that the protocols describe the accessed data elements. One protocol/vendor may require several attributes to express a *Capability* while another may express it as a complex string. A *ServiceElementType* is the intersection of a set of *Capabilities,* access methods and data formats for a ServiceElement. The 'keys' are:

- Vendor
- Model
- Firmware
- Software version
- Sometimes (feature set) that is part of each *ServiceElementType*.

[3] It is the case that there are many attributes that are also associated with a ServiceElement from calculated operational state, capacity, and external factors. In this discussion the point is to distinguish a type of a system from other types of system that are not determined by state or external attributes but by how they respond to external commands.

- Where available boot image name.
- Mode variations - in some cases, the configuration of a system will add or remove other configuration or monitoring attributes. The presence or absence of some attributes is an indicator that the service element is of the same type as another, but is configured differently.

When possible, additional information related to the type of system is also associated with the ServiceElement by virtue of its assigned ServiceElementType:

- Vendor specific sub-type as would be the case with a particular interface plug-in module.

Each instance of a *ServiceElement* has one, and only one, *ServiceElementType*. The *ServiceElementDiscoveryManager* will match the information retrieved from a *ServiceElement* (e.g, vendor, model, software revision) and set its *ServiceElementType*.

Incomplete Information and ServiceElementType Assignment

Initially, Integer will come across *ServiceElements* for which there is insufficient information to make a determination as to *ServiceElementType*. Integer uses all the information available and where there is doubt, Integer will use a generic *ServiceElementType*, flag the discovered element, and let the operator know so that additional information may be entered by the user to create a more refined *ServiceElementType* for this element. With this approach, subsequent discoveries of this and other *ServiceElements* of this type will be assigned the more refined *ServiceElementType*. The more that is known about the details of a device the better Integer will be able to effectively manage and correctly configure it.

What is a ServiceTechnology[4] and how is it Different from a ServiceElement?

ServiceElements do the work. They either directly perform a function like forwarding packets, store information, or contain other ServiceElements that do work in the sense a chassis might contain cards. In contrast, a *ServiceTechnolgy* is an abstraction that represents a tree of increasingly more specific detail of the components of the *ServiceTechnology* all the way to specific management objects appropriate to one and potentially many *ServiceElementTypes*. Additional characteristics:

- The components placed in a *ServiceTechnology* must be related to that technology. For example one would not place attributes associated with a relational database in with routing technologies.

---

[4] Integer uses *ServiceTechnology* instead of Service to make the distinction between even more abstract things like a payroll application or a student information system, from the technologies that implement those end-user visible services. In the Integer system, these higher-level concepts are represented by what are called *BusinessServices.* Subsequent releases of Integer that focus on fault, performance, capacity planning and other management functions, will make more use of the *BusinessService* concept.

- They often depend on the proper local functioning of other technologies. For example a name server is dependent on routing working where it is located.
- *BusinessServices* combine any number of *ServiceTechnology* elements to define something users would understand like email, a payroll application, or learning management environment. By creating these relationships, users will be able to quickly understand how individual *ServiceElements* of many different *ServiceElementTypes* are used to support these complex applications. Future releases of Integer will leverage these facilities.

See Appendix E - A Complex Technology Service Hierarchy - Small Portion for an example. It is important to note that the definition of a *TechnologyService*, like many other aspects of the system are user configurable. As much default information will be configured in each release as time and resources permit. Users can add to these or modify as needed.

InterDeviceLinks

InterDeviceLinks are created by the *TopologyManager* each time it can identify an inter system connection between a *ServiceElement* in one system and a *ServiceElement* in another system. These links can be of several different types, most commonly they will identify an adjacency between systems at the layer 2 or 3 level. They can however be used to express a special relationship between devices as would be the case with CDP or with VSS pairs. New types can be defined as needed. These links are always created in pairs, one for each direction. Note that these links are not full paths as between IP routed systems. The links are combined to create a *Path* in the Integer system.

The Discovery Service and Managers

This section describes the main components of the discovery system and their relationship to each other including how they would operate in a distributed environment.

Discovery Service

The discovery service is global in that there is only one running for each Integer installation. It is responsible for reading the discovery rules and controlling the behavior of the *ServiceElementDiscovery* and *TopologyManagers*. Unlike services such as the *DiscoveryService*, managers in general, and these two in particular, may have multiple instances running on an *Integer installation* at one time. See appendix D, Discovery in an Distributed Environment for additional information.

Based on discovery rules stored in the database, the *DiscoveryService* will kick off one or more *ServiceElementDiscoveryManagers* and/or *TopologyManagers* at a time, not all of which are required to be on the same system as would be the case in a large installation where several Integer servers have been deployed and configured to support the *DiscoveryService*.

The *DiscoveryService* also performs a number of other functions:

- Provides the Global SNMP Credentials List to subsystems.
- Future releases may require global credentials for other non-SNMP discovery approaches.
- Status on discovery rules processing.
- Status on active *ServiceElementDiscoveryManagers*
- Status on active *TopologyManagers*

Service Element Discovery Manager

Instances of the *ServiceElementDiscoveryManager* are relatively short lived in that they are started by the *DiscoveryService* and assigned a single subnet. The following is a brief list of *ServiceElementDiscoveryManager* functions. Later sections have additional detail:

1. Find systems on the network it has been assigned.
2. Create new instances of these systems in the database if they do not already exist.
3. Identify the *ServiceElements* that comprise the system.
4. For each service element discover and assign its *ServiceElementType* wherever possible.
5. Discover services running on the *ServiceElements* important to network operations as defined in the configuration of the *DiscoveryService*.

When their assigned functions are complete, *ServiceElementDiscoveryManager* instances terminate. With this approach it is possible to have multiple instances running on separate subnets at a time to improve the overall performance of the system, reduce network traffic and allow each instance to be tailored to the specifics of the subnet on which it operates (e.g., localized credentials for the network).

Service Element Discovery Manager Functions

ServiceElementDiscoveryManagers provide the following functions to the system:

- Identification of those systems (*ServiceElement*) present on the network at the time they are executed. In this case that means that they responded to an SNMP request for 6 items from the *Systems Group*.
- Identification of a system's Vendor, Model, Firmware and software revision.
- Creation of *ServiceElements* in the system for each main and sub component found.
- Based on information collected from the *ServiceElement*, each *ServiceElement* created in the system will have a *ServiceElementType* associated with it that contains additional information about the *Capabilities* of the component.
- Does the service element provide routing, DNS or other services? See the Service Discovery section that follows
- Identification of a system's containment hierarchy - a goal of Integer is full *service management*. This means that Integer must not only know what a device is but also all the child hardware and software components it has.
- For each sub component (also called a ServiceElement), the process is repeated.

- Flagging things it does not know. Integer can be configured with new information in the field which can be shared with other Integer installations if desired. It is expected that in the early releases of Integer, many *ServiceElementTypes* may be unknown to Integer. In those cases where it does not 'know', or can not discover what is the correct *ServiceElementType* to assign to a *ServiceElement*, information will be collected and flagged so that Integer can be 'taught' how to recognize the *ServiceElementType* in the future.

ServiceElementDiscovery Managers Retrieval of Interface Information

Because networking and network connectivity is central to service delivery, the *ServiceElementDiscoveryManager* will retrieve interface and address information from each *ServiceElement* instance (even when a full topology discovery is not requested). Integer will use the standard and vendor-specific containment discovery approach for this function, but will augment it with standard and vendor-specific objects that focus on interfaces and related information such as the Interfaces MIB. This process will result in creation of additional objects (*TopologyElement*) that detail information about interfaces such as IP address, MAC address, etc. Note that interfaces (both physical and logical) are *ServiceElements* in the Integer system, *TopologyElement* instances contain additional information for those *ServiceElements* that have address related information.

Technology Service Discovery

In addition to discovery of the containment hierarchy of a device and details of its constitutive *ServiceElements*, the *ServiceElementDiscoveryManager* attempts to identify services that are generally associated with how networks move packets from one place to another. These services include:

- Layer 3 routing like BGP and OSPF
- Layer 2 elements like VLANs
- Network services such as DNS
- Firewall functions
- Load balancing
- NAT

The information about services is saved and then provided to the Topology manager if network topology discovery has been selected as part of the *DiscoveryRule* (see Appendix F - Technology Service Discovery).

A *ServiceElementDiscoveryManager* instance is not responsible for determining differences between information collected in one run versus a subsequent run. It simply creates new objects when a ServiceElement does not yet exist in a system. The InventoryManager is responsible for determining change - based on *InventoryRules*.

Topology Manager

Instances of the *TopologyManager* most closely resemble discovery systems that some may be familiar with. It is responsible for learning network topology. It also extends in a number of dimensions, details follow:

Address Range Calculation

A number of techniques including PING have been used over the years in these systems to determine which systems on a subnet are reachable. This approach can be inefficient. Integer uses a library that calculates the range of possible IP addresses based on a network and mask or the current CIDR notation. This information is passed to each ServiceElementDiscoveryManager as it starts each subnet.

Layer 2 and 3 Discovery

When the configuration of the discovery service indicates that a layer 2 and 3 discovery is to be performed, the DiscoveryService will delay the initiation of the *TopologyManager* on a subnet on which any *ServiceElementDiscoveryManager* is active. The objective is to reduce the probability of the Integer system having more than one portion of the system query a device at one time[5].

The *TopologyManager* starts in the network indicated in the *IpTopologySeed* augmented with information from the *ServiceElementDiscoveryManager* that just completed the discovery of *ServiceElements* on that subnet. Not only is information about L2/3 forwarding systems provided, it is also supplied with information about 'path modification systems' see below.

Path Modification Systems and Other Technologies

As part of the pass over a subnet, the *ServiceElementDiscoveryManager* can be configured to collect information about specific technologies that are of interest. Initially, Integer will be configured to examine systems for forwarding functions for layer 2 and 3. This information is passed to the *TopologyManager* to improve its effectiveness when learning about the network topology.

Integer defines systems that provide services such as NAT, load balancing and firewall functions as path modification systems. They can be stand alone systems or the services may be integrated with L2/3 systems. The objective of this phase of discovery, if selected by the user, is to provide an overlay layer of information about how the flow of packets through a network is altered by these systems. This is a challenging function so Integer may have to provide incremental additions of function over a series of releases.

---

[5] New *ServiceElementDiscoveryManagers* are initiated via the *DiscoveryService* as the *TopologyManager* discovers new subnets.

Inventory Manager

The *InventoryManager* is a specialized reporting and analysis system that also provides alarms based on one or more events as defined in an *InventoryRule*. An *InventoryRule* can be global, that is applies to the entire Integer database or it can specify a subset. The criteria used for selection of *ServiceElements* covered by an *InventoryRule* are:

- Location[6].
- Criticality of the *ServiceElement*.
- *Technology* such as routing, name service, database service, etc.
- Whether there have been changes or not.
- Systems found since the last discovery run.
- Systems not found or changed since the last discovery run.

A number of controls are included in an *InventoryRule:*

- Number of Discovery runs that must take place before a system is counted as new/added. This can avoid churn if sections of the environment are under construction and change.
- Number of Discovery runs that must take place before a system is deemed 'missing'.
- Actions to take when a system has been added or deemed to be missing or changed (e.g., notify, or notify and report).
- Systems to exempt because of administrative state (e.g., ifAdminState = down).
- Determining when something has changed - due to the differences that exist from one *ServiceElementType* to another, Integer allows three different ways to define a changed *ServiceElement*:
  - If the *UniqueIdentifier* of the *ServiceElement* changes.
  - If one or more of the *additionalAttributes* in the *ServiceElement* collected by the *ServiceElementDiscoveryManager* changes.
  - If a combination of both the *UniqueIdentifier* AND one or more of the *additionalAttributes* changes.

Note that change in the Integer first release context is confined to changes to data visible to the *ServiceElementDiscoveryManager* and the *TopologyManger* instances. In future releases this facility will be extended to perform other types of change reporting for example, configuration changes.

Distributed Operation

See Integer Distribution Model Document details the methods used for distributed operation. This section takes a closer look at one scenario in the context of the *DiscoveryService*. In this example, there is a large network, the central campus with

[6] Location, criticality and technology are examples. The InventoryRule will have a Selection instance associated with it that will function as selections do in other places, except as otherwise noted in the class definition.

many subnets that is contiguous (that is within a private address space). The network also includes two remote sites connected to the main network via the Internet.

Because of the size of the large private subnet - the central campus - Integer has been deployed in a distributed fashion. There are two clusters of servers in the data center. Each of these backend servers have been assigned a range of subnets for which they are responsible. The central server is where the *DiscoveryService* operates. It will cause multiple instances of the *ServiceElementDiscoveryManager* and the *TopologyManager* to be instantiated on each of the backend servers as needed. The *DiscoveryRule*, and the users that create the *DiscoveryRule*(s) do not have to specify this information. The distribution layout is managed by the Integer system administrator.

In each of the networks that are remote from the central campus has a backend server deployed. They have been configured to communicate with the central server over a secure connection. See the diagram, Discovery in a Distributed Environment in Appendix - D at the end of this document for more details.

Multiple Discovery Rules and Overlap

The nature of networks is such that it may not be possible to accurately predict if different discovery rules overlap in a network. For example when two different discovery rules that start on subnets several hops from each other but which have a large *radius*[7]. If the *DiscoveryService* has the *excludeOverlappingSubnets* attribute set to true, then when *DiscoveryRules* run separately would cause the same *ServiceElements* on a subnet to be discovered multiple times, the system will skip the overlapped subnet. If *excludeOverlappingSubnets* is not true, then the new information will overwrite the old. The *TopologyManager* checks each subnet before starting.

User Interaction with the Discovery Service

While integer is flexible and can be extensively configured to meet a wide variety of conditions, some may want to start using the system with the default information supplied without adding additional system information. This simplified mode of operation is supported. The following sections describe:

- User Configured Data
- Discovery Control and Defaults
- Data Initially Supplied with Integer

User Configured Data

This discussion assumes that the data initially supplied with Integer is in the system. Discovery effectiveness and coverage may be increased through the addition of more data and its configuration as described below:

---

[7] *Radius* is the number of hops from the identified starting subnet in an *IpTopologySeed* that the *TopologyManager* will hop.

- *SNMPVendorDiscoveryTemplate* - these templates are used by the discovery to determine model, software revision and other information based on the System Group or other SNMP OIDs specified. This information is used [by the *SNMPVendorContainmentSelector]* to tell the system how to discover the containment hierarchy for the system (like cards, ports, disk, memory, etc). Absent this information a generic approach will be used (Entity MIB) but many systems do not support this (or support the objects at know locations). The result will be a discovery only of the initial information and interface information if available.
- *ServiceElementType* definitions - How many of these have been defined for the systems that are encountered by discovery. For example if there are none, then Integer will still be able to discover a top level device from the SysOID and determine its vendor. The result of this will be a generic icon for something that exists but for which we do not have details. This could happen at the top level as described above. It could also happen within the containment hierarchy. Integer will use what information is available and show that it is not complete. For example, if we know that we have discovered a card with ports but can not learn anything beyond that we will use a generic icon for that.
- *ServiceElementManagementObjects* - These are the objects imported into the integer system that represent the protocol-specific data used to communicate with *ServiceElements*. They could be SNMP based (the only supported protocol for the initial system) or any of a number of standard methods used for monitoring and control including via an interactive terminal session. When imported into Integer it is possible to add metadata to them that allows more advanced functions such as understanding what to query a device for to learn if it is a load balancer or where to retrieve a serial number or which objects are usage counters verses fault counters.
- *Capabilities* definition and association with *ServiceElementManagementObjects* - Any cross vendor system must have a way to abstract general concepts such as a VLAN tag from the specific instructions sent to a device to configure the tag, or to retrieve the value of VLANs from an interface (including how to tell the device which interface it is interested in). *In the Integer system, Capabilities provide the way different ServiceElementTypes can be treated as the same thing even when they have potentially no overlapping ServiceElementManagementObjects in common. In the 'real world', there is a strict connection between a ServiceElementType*[8] *and the ServiceElementManagementObjects it can support. Capabilities abstract the details so that if a VLAN needs to be created on an ethernet interface it may be done on different ServiceElementTypes even when the configuration protocols and data they carry are different. Capabilities, and ServiceElementManagementObjects by their nature are very granular, for example a VLAN tag number.* Specific instructions can vary based on management protocol/access method, data formats, vendors, models, software

[8] Integer does also support the idea of a generic *ServiceElementType*. These exist when Integer does not know enough to definitively say what something is, but has enough data so that a reasonable inference may be made. For more on how this works, see Appendix

revisions and other factors. A *Capability is* the abstraction that achieves this function in Integer.

- *ServiceTechnologies* continue the abstraction from *Capability* objects so that the system has a convenient way to operate on like concepts even though they are implemented in different ways. For example, if one wanted to discover all servers that are related to an Email system, one could select that *TechnologyServices* that contain details related to a mail system. For example the DNS, mail servers, dedicated virus scanners, etc.

Discovery Control and Defaults

There are several configurable elements that control how the *DiscoveryService* operates:

Discovery Service

Attribute	Description	Default
Allow overlapping subnets	If discovery rules exist that when run separately would cause a the same subnet to be discovered multiple times then if this value is true, then excludeOverlappingSubnets will cause the system to skip the overlapped subnet.	No
Discovery Mode	The mode that the discovery system operates in. Two are initially supported, concurrent and sequential. In sequential mode, the topology manager(s) are not started for a discovery rule until all ServiceElementDiscoveryManagers have completed their work for a given IpTopologySeed. In the case of a topology that discovers a new subnet, then the ServiceElementDiscoveryManager will be started after the topology manager 'exits' the subnet - that means that the ServiceElementDiscoveryManager has completed all the elements in the hostRange. Note that it is still possible for the ServiceElementDiscoveryManager and the TopologyManager to 'collide' on a single system (e.g., a router) if multiple seeds are active at one time. The user has two ways to control this: Scheduling and control of the SNMP requests to a single device	Sequential
Community String List	An ordered list of global SNMP community strings that will be used to discover a previously unknown device. This list is used after local	Public
v3 Credentials	SNMPv3 credentials (TBD)	None
Alternate Port List	An ordered listing of alternate ports (other than 161, the default) that can be used for SNMP-based discovery. Users can change the default if needed.	161

Discovery Rules

Attribute	Description	Default
Calendar Policies	Each discovery rule will have at least one, and potentially many *CalendarPolicies* associated with the discovery. These policies are used to initiate the discovery process at controlled times.	Daily at midnight
IpTopologySeeds	A listing of the *IpTopologySeeds* to be used with this discovery rule. Multiple seeds may be used to give operators control of the extent and type of discovery that is run.	3 Hops from main Integer host

IpTopologySeeds

Attribute	Description	Default
Network Exclusion List	A set of networks not to be included in the discovery even if they are within the radius of the discovery.	None
Gateway Exclusion List	A list of gateways to exclude. This will preclude discovery of all information from a specified gateway and continuation beyond that gateway even if the radius indicates the topology manager is to do so.	None
Starting subnet for the discovery process	If desired, the discovery can start from a specified subnet.	None
Radius	How many hops from the starting subnet the discovery process is to go.	3
Discovery Order	List of SNMP protocols in the order that they are to be tried. That is, SNMPv2, SNMPv3, SNMPv1	SNMPv2
Technologies to include or exclude.	Integer will discover technologies such as routing, load balancing, etc. For this to function, the system must be configured with the information for the discovery system to make these determinations.	Routing Load Balancing NAT Firewalls
Include/Exclude	For the technologies identified in the include/exclude list, this indicates whether they should be included or excluded.	Include

Attribute	Description	Default
SNMP Timeout Service Element Discovery Manager	The time out value for SNMP messages used by the Service Element Discovery Manager in this IpTopologySeed.	None
SNMP Retries Service Element Discovery Manager	SNMP retries for messages sent by the Service Element Discovery manager with this IpTopologySeed.	None
SNMP Timeout Topology Manager	The time out value for SNMP messages used by the Topology Manager in this IpTopologySeed.	None
SNMP Retries Topology Manager	SNMP retries for messages sent by the Topology Manager with this IpTopologySeed.	None
Initial Gateway	This is a more precise way of defining where you want the discovery to start on a subnet.	None
SNMP Local Credentials	Users can specify overrides to the Global SNMP Credentials set up in the Discovery Service. Multiple sets of credentials can be associated with each IpTopologySeed. This allows a separate set for each subnet if desired.	None
Discovery System	In a distributed Integer environment, it is possible to have the DiscoveryManager controlling ServiceElementDiscovery and TopologyManagers on remote systems. Users tell the integer system where they want the discovery to be initiated from with this attribute.	None

Service Element Type - Additional Controls and Information

Attribute	Description	Default
Service Element Type Override(s)	Some vendors/software versions have difficulty processing SNMP messages, especially if a large number are sent at once. This allows Integer to control the maximum rate that messages will be sent to specific types of systems based on ServiceElementType, how many retries for each message and what the time out value, and maximum number of outstanding messages should be. These settings override settings in the time outs and retries specified in the topology seed for the service element types identified.	None

Attribute	Description	Default
Additional Attributes	If desired, for each service element type, additional attributes the Service Element Discovery Manager can retrieve may be specified. For example, when Integer collects information from an interface, it will by default get information such as ifName, ifSpeed and ifDescr.	None other than those listed to the left.

Additional Data Initially Supplied with Integer

This list will change and grow over time. Also note that while many SNMP MIB Modules have been added to the system as a convenience, the majority will not have been configured/connected with capabilities and thus will not be used in operation until this is completed. The following is a list of MIB Modules that are part of every build. A full list, including vendor-specific modules will be in the manifest associated with releases of Integer as it is too long to include in this document.

- RFC1065-SMI
- RFC1155-SMI
- RFC-1212
- RFC1213-MIB
- SNMPv2-SMI
- SNMPv2-TC
- SNMPv2-CONF
- SNMPv2-MIB
- IANAifType-MIB
- IF-MIB
- IP-MIB
- SNMP-FRAMEWORK-MIB
- ENTITY-MIB.my
- HOST-RESOURCES-MIB.my
- CISCO-SMI.my
- CISCO-ENTITY-VENDORTYPE-OID-MIB.my
- CISCO-TC.my
- CISCO-PRODUCTS-MIB.my

Reporting Service

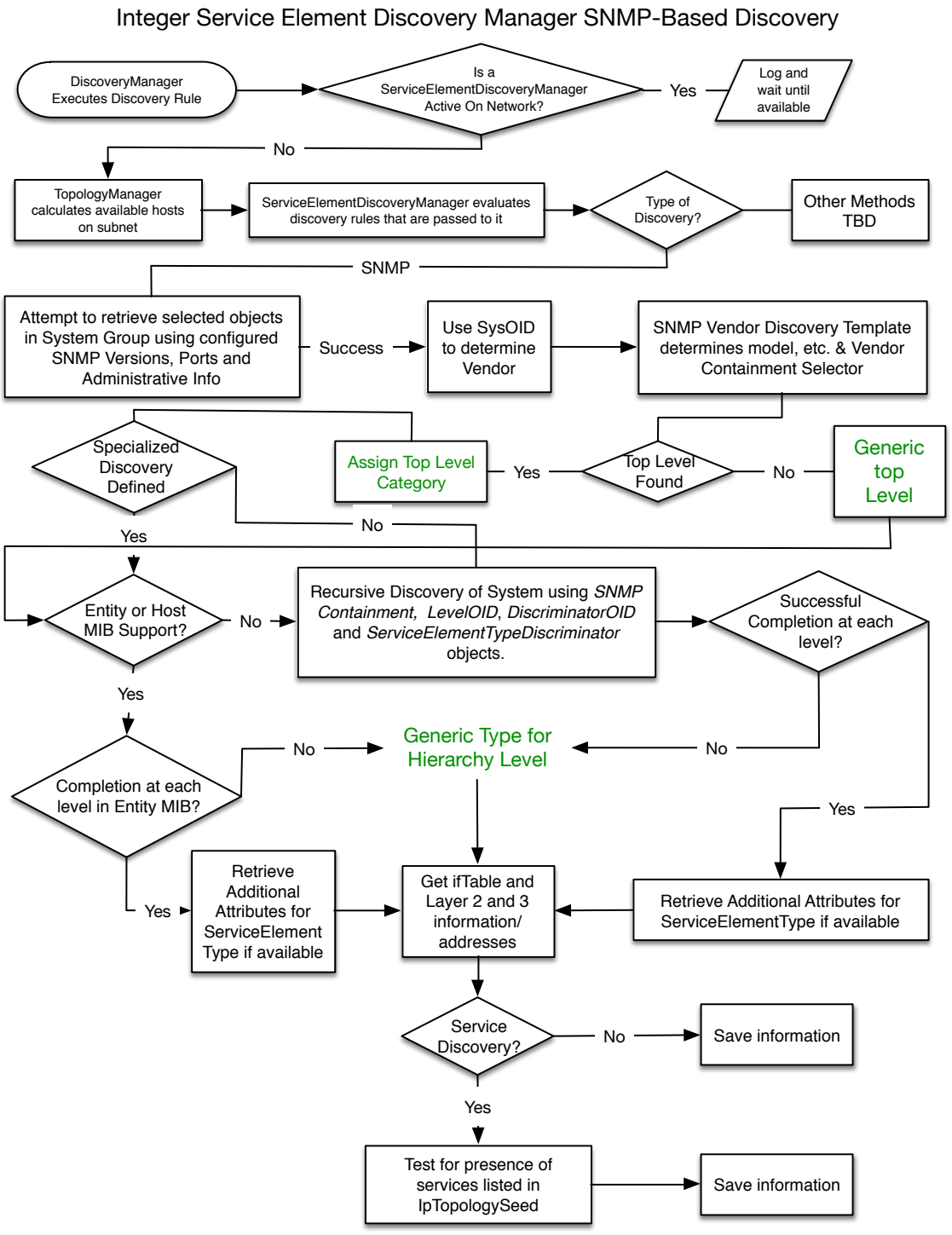
The *ReportingService* in the first release is limited to supporting discovery functions and providing output needed by a variety of other systems manual and automatic. As with other elements of the system, the work of producing work may be distributed to other Integer system. The following table describes the information a user would provide to produce reports.

Reporting Service Controls

Attribute	Description	Default
Name	User assigned brief name of the report.	None
Description	Description of the report.	None
Report type	Type of information contained in the report such as discovery deltas, other types will be defined as new functions are added to the system.	None
Report format	Format of the report, such as tabular, graphic, etc.	None
Email list	Users can have lists of users that will receive email from the system if desired.	None
Retention	How long generated reports should stay in the system.	90 Days
Networks	Users can specify specific networks to include or exclude from the report.	None
Excluded/Included ServiceElementTypes	Users can specify what types of service elements to include or exclude from the report	None
Export format	If the report is to be exported, what format is used?	JSON

Appendices

Appendix A - SNMP-based ServiceElement Discovery Overview

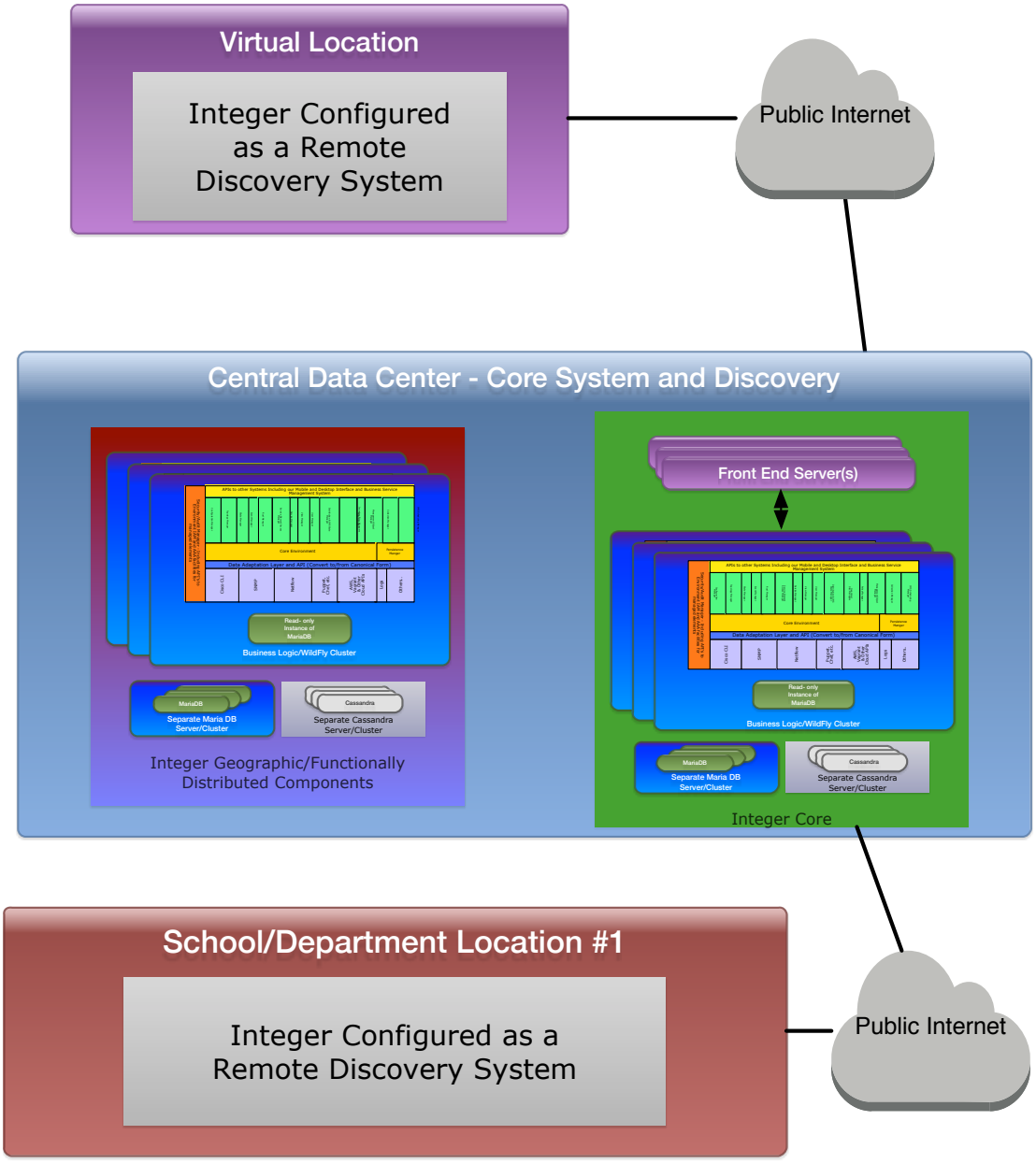


Note: There is always the possibility that Integer will not have the information to be able to deduce what ServiceElementType to assign to an element in the discovery process. When this occurs we will use generic types appropriate to the level in the hierarchy if possible.

Appendix B - ServiceElementType Discovery Process

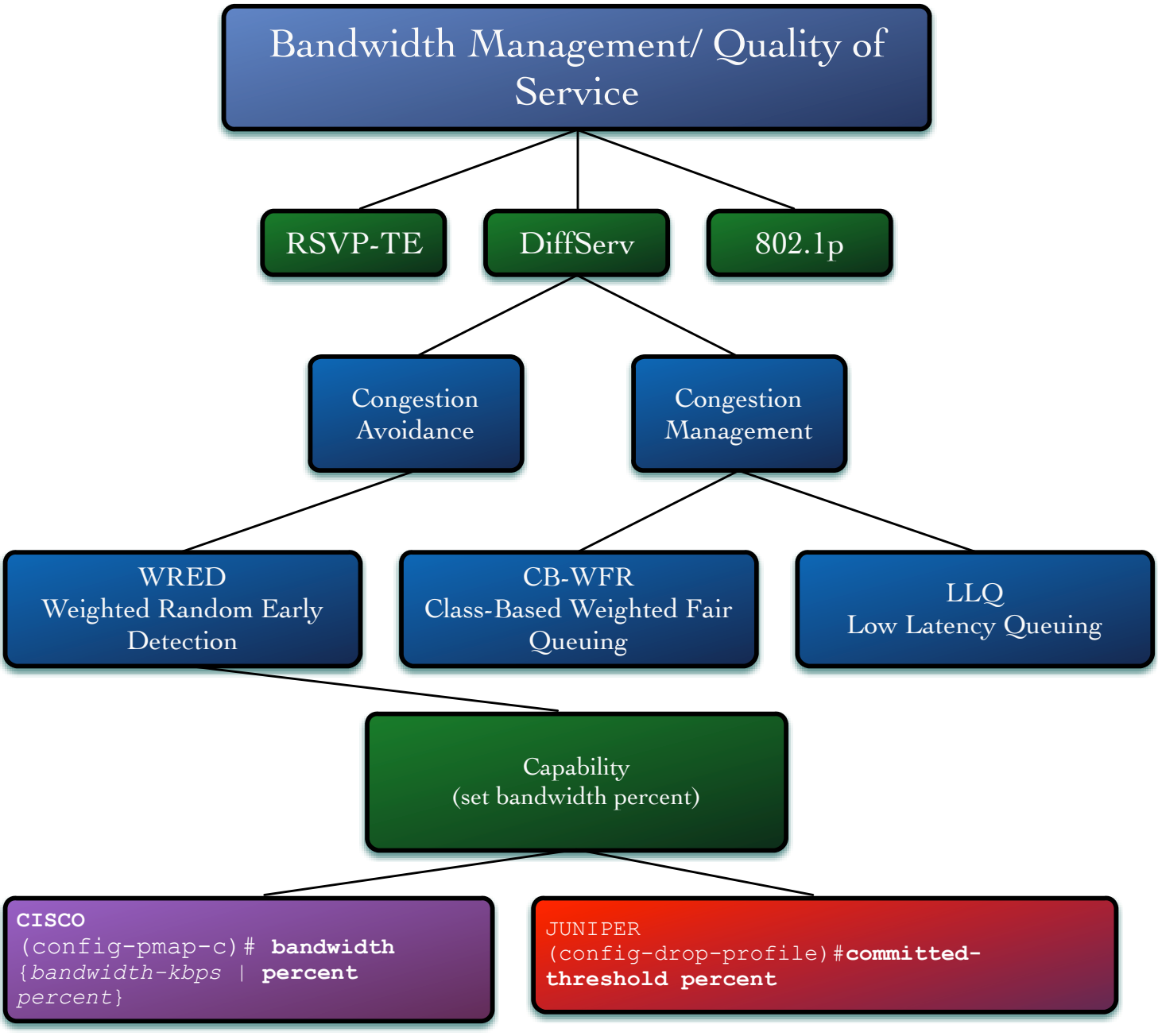
Appendix C - Topology Discovery

Appendix D - Discovery in a Distributed Environment

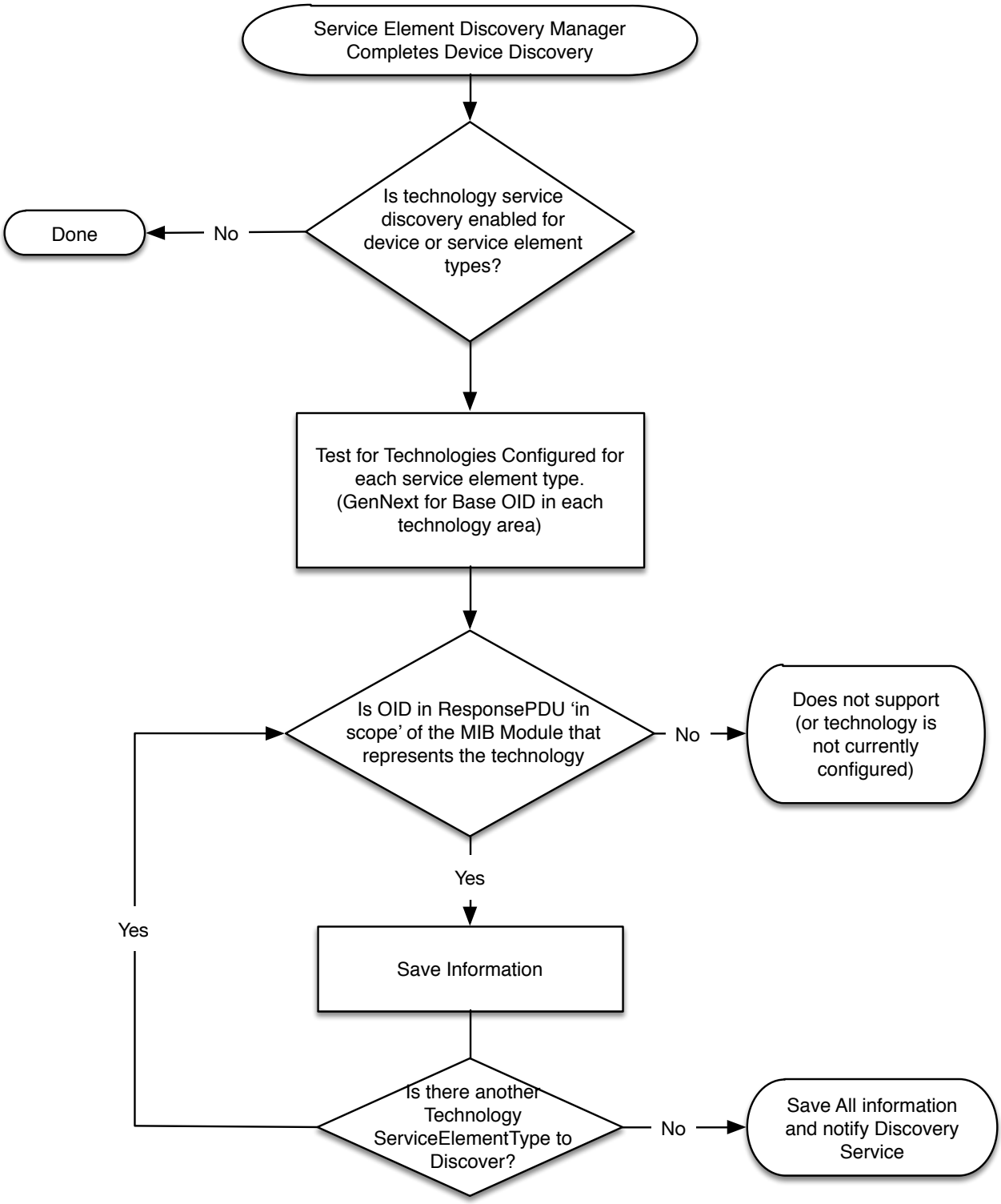


Integer may be deployed in locations accessed via either internal networks or the public Internet. How each installation is configured determines the functions provided, the images are the same. The *DiscoveryService* will provide the discovery rules to the distributed components as needed and the information they collect will be sent to the Core System. For additional details on distributed operation, see the Integer Distribution Model in Appendix G.

Appendix E - A Complex Technology Service Hierarchy - Small Portion



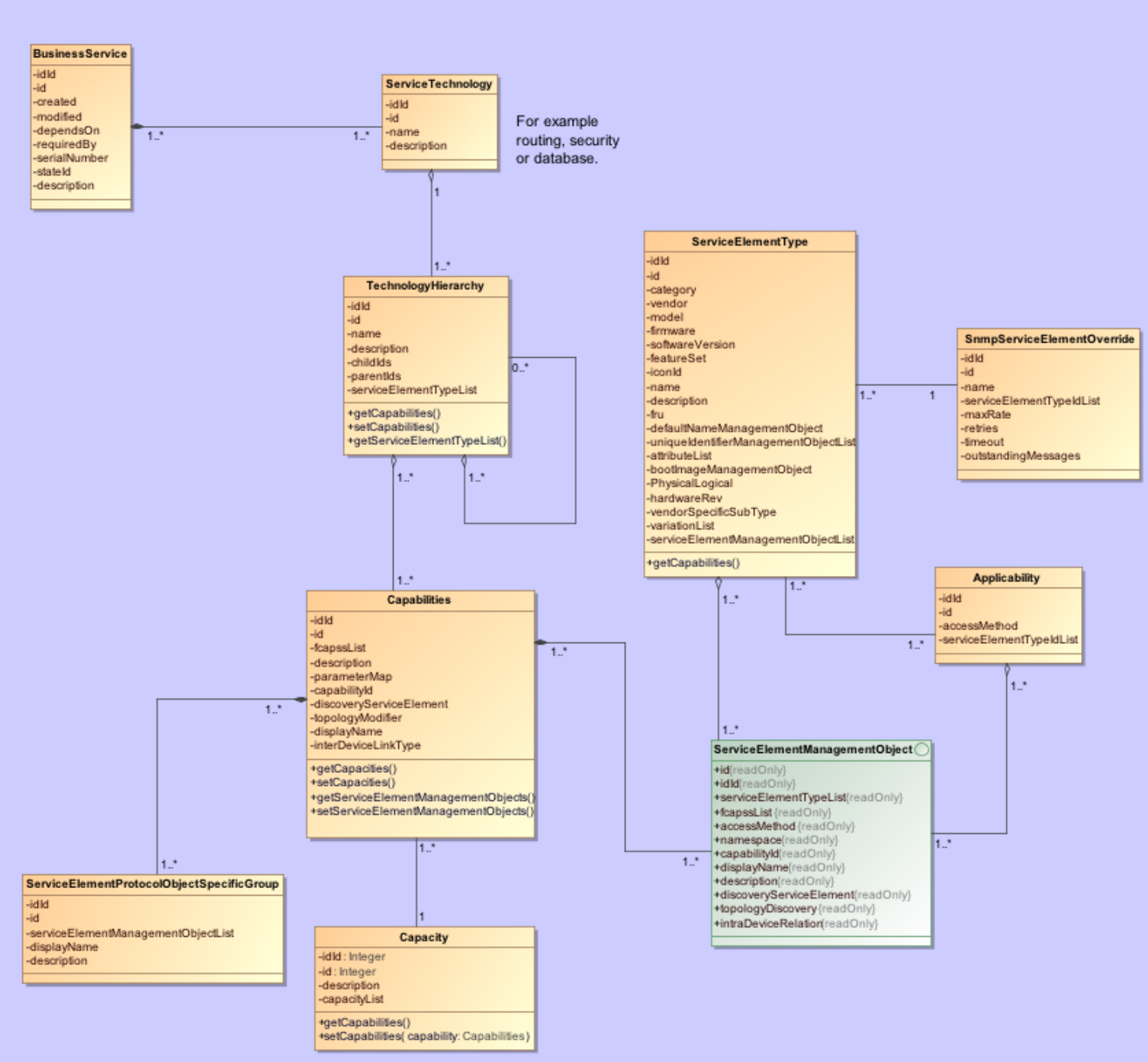
Appendix F - Technology Service Discovery



Appendix G - Key Relationships

The Diagram below shows the important relationships between:

- *ServiceElementTypes*
- *Capabilities*
- *The TechnologyTree*
- *ServiceElementManagementObjects*



Appendix H - Reference Documents

1. Integrated Management System - Project Integer
2. Integer First Release Functions and Objectives
3. Integer Distribution Model
4. Integer Access Control Model
5. Alternate Login
6. General User Interface Objectives and Requirements
7. Integer Selections