Technical White Paper

Perform Service-Oriented Orchestration with Cisco Process Orchestrator

What You Will Learn

Cisco® Process Orchestrator 3.0 introduces a new feature set that provides service-oriented orchestration and enables a shift away from traditional run-book automation and IT process automation. With this new approach, automation aligns with the high-level services provided by IT and models the way that a high-level service is supported by a topology of lower-level services, systems, and devices. Planning for services and their desired states is the initial step in automation design. The next step is definition of process actions for these services and then implementation of specific process workflows that traverse these services to act on lower-level elements. This enables a declarative approach to automation, focusing on the result desired rather than how it is achieved.

The capabilities augment and complement traditional orchestration definitions and approaches, so that Cisco Process Orchestrator provides both service-oriented and process-based orchestration. This document focuses on the incremental, differentiated capabilities that service-oriented orchestration adds to Cisco Process Orchestrator.

Overview

Cisco Process Orchestrator is an advanced orchestration engine belonging to the run-book automation (RBA) or IT process automation (ITPA) class of products. Traditionally, tools in this category focus on a sequence of IT processes that implement automation. The process is the focal point of automation. Processes act on lower-level IT elements such as devices, servers, and specific tools. The set of elements on which automation acts is typically delivered in the product through adapters connecting to various layers of the IT technology stack. IT, however, focuses on services that provide value to the business, which are much higher in the stack. The inability of RBA and ITPA tools to act on the business-level services in the environment becomes an inhibitor to delivery and creates a poor abstraction for users.

Service-oriented orchestration provides the agility to model and act on IT services. These features make creation of orchestration active and dynamic and allow new, higher-level services to be defined in the system and deployed quickly. After new types of services are defined, the organization can create instances of those new services. Using events, automation can detect patterns in these services, enabling policy-based automation.

Service-oriented orchestration incorporates several industry trends to synthesize a fresh approach to orchestration:

- The service modeling capabilities of a service catalog are now available in the orchestrator layer. Cisco Process Orchestrator provides fluid mechanisms for exchanging service information with Cisco Prime™ Service Catalog, advancing the integration of these systems.
- The feature delivers many of the capabilities of object-oriented design and programming to ITPA. The shift from traditional orchestration to service-oriented orchestration is similar to the shift from procedural programming to object-oriented programming. Today, almost all programming is performed using object-oriented languages, and object-oriented design has transformed the high-tech industry, increasing productivity and quality. Service-oriented orchestration has the same promise.
- The IT Infrastructure Library (ITIL) prescribes a service-centric approach for IT. Configuration management databases (CMDBs) model IT services and their relationships to other IT assets. Service-oriented automation allows automation to be guided by a model of current and potential IT services with awareness of their relationships and interdependencies. In this way, the principles of service modeling in Cisco Process Orchestrator are essentially the same as those for modeling services with ITIL and CMDBs. Also, although Cisco Process Orchestrator can integrate with an available CMDB, a CMDB is not needed to enable orchestration.
- The feature aligns with industry standards such as the Distributed Management Task Force (DMTF), Common Information Model (CIM), and the Topology and Orchestration Specification for Cloud Applications (TOSCA).
- Model-based automation using script-based tools is becoming popular, especially for configuration management. Cisco Process Orchestrator combines the capability to model services with the openness to integrate with these tools to build on their strengths. Moreover, the feature allows model-based orchestration atop traditional tools to integrate the full power of model-based approaches with other IT tools.

Service-oriented orchestration allows automation to focus on higher-level IT services, possibly services specific to the customer's business and unknown by Cisco when the product is built. User interaction shifts from lists of processes to services and what the user can do to them. Instead of having to first decompose automation into a sequence of processes, high-level services are decomposed into their components, and then the actions possible for each service are defined in automation. This inversion in approach is simple, yet powerful. Cisco Services, partners, and customers can model services and extend others' service models without coding. Extensions and automation can be packaged, shipped to customers, moved from development to test to production environments, versioned, and upgraded.

This approach delivers several main benefits:

- Cisco Process Orchestrator combined with Cisco Prime Service Catalog facilitates ease of use: The Cisco Prime Service Catalog Adapter simplifies the exchange of service requests and service items to and from the orchestrator. Users can easily create targets from incoming service requests. Cisco Process Orchestrator helps push service items back to the catalog as needed and allows users to pick the right moments in a flow to synchronize data. Use of an active catalog connection in the Cisco Prime Service Catalog Adapter optimizes the user experience. With awareness of the Cisco Prime schema, the Cisco Prime Service Catalog Adapter easily reads, writes, and creates service items. When a service item definition is created in the catalog, integration can be achieved rapidly. The property browser exposes the catalog definition to enhance ease of use. "Get Service Item" properties can be followed by "Update Target"

to save properties in a single step. Users can write directly through the “Create Service Item” or “Update Service Item” activities using property references.

- Workflows are simpler and more readable.
- Data defines the desired services. The service instance guides automation to achieve the desired state. The approach separates the desired state from the implementation process, separating the “what” from the “how.”
- The approach acts on the higher-level service rather than its technology elements. Services can span tools. Workflows can navigate service topologies to lower-level elements on which they act.
- Operation views of automation by service are provided.
- Organizations can monitor the environment in conjunction with the service definition and bring both in line with policy.
- Federated storage can be used. Objects and relationships can be pushed to and from service catalog, CMDB, and service assurance tools when needed.
- Automation is easier to extend and customize.

Cisco Process Orchestrator Features That Support Service-Oriented Orchestration

Several Cisco Process Orchestrator capabilities combine to support service-oriented orchestration:

- Service instances are targets. A target type defines a target.
- Target types support inheritance, so that you can extend a general type for a specialized need.
- Target types have extensible lists of properties, including field-customizable default values.
- Relationships allow modeling of topologies of lower-level services assembled to offer higher-level services.
- Processes provide actions that can be run against targets. You can view a target to see what actions are possible or what automation is being performed against it.
- Targets and target types have events. These events can be internal process events or open Advanced Message Queuing Protocol (AMQP) events. Triggering processes in response to patterns in underlying processes provides policy.
- Service instances can be extended by services, partners, and customers. The content, not the platform, defines a service model. The platform is open for modeling any IT service topology in the automation content. For example, some types, actions, and properties may come from a packaged product, some from team 1, some from team 2, some from a partner, and some from a customer. Using this capability, you can assemble a complex solution from parts. Each author controls the version and lifecycle of the elements that he or she delivered, so each author can provide upgrades.
- Service instances use a consistent API, even if elements come from different authors.

Target Types

A target type allows you to define a new service. All new targets are created based on a target type. You use the target type to model the services on which you want to act. Target types can be bound to actions delivered through Cisco Process Orchestrator processes. For example, you can provision an application described by a target type. Target types define inheritance, properties, relationships, and process actions.

Inheritance

Through inheritance, a target type can contain all aspects of another type but can provide an extension through a more specialized type. For example, a cat is a specialized type of mammal, and a mammal is a specialized type of animal. A cat is a mammal and hence is an animal. If a mammal generically can perform an action such as wake up, the cat can also perform that action, but the cat can also do additional things such as meow and purr. If you define a property of a mammal, such as weight or body temperature, a cat will also have that attribute, but you can also add properties such as breed, Siamese, tabby, etc., that apply to cats but not to all mammals.

This concept is very useful in real-world modeling of types of IT services and can greatly simplify implementations. Inheritance can allow common aspects of a collection of types to be defined once in a shared type and then inherited, without the need to repeat the common implementation in each type in the collection. The mechanism avoids duplication of properties and processes, making the automation easier to maintain.

Also, it is often simpler to implement something new by starting with something for which automation already exists and extending the automation for a new specialization. This approach allows you to focus on only what is new and unique, rather than all the aspects, including those already addressed by existing automation.

Note: If a type “has a” relationship rather than “is a” relationship, you should use target relationships, not inheritance.

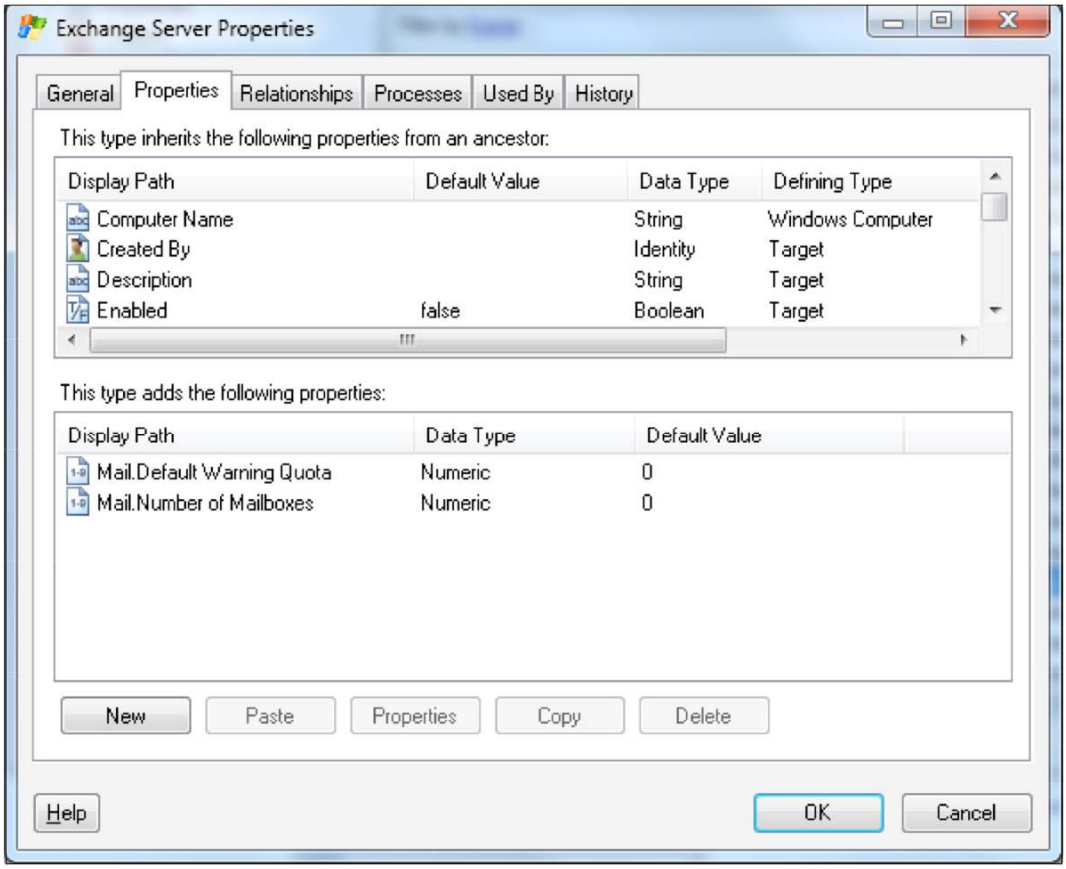
Target Properties

Target properties allow storage of information for targets created from a target type. For example, a target property can store network device discovery information, such as information about the operating system, version, optional cards and modules installed in a device, serial numbers, and configuration. Network automation can then use this information in its workflows. Target properties can be queried in target groups, in triggers that watch for patterns in the data, in Find Targets activities in a process workflow, or in target selection algorithms in processes, allowing you to act on collections of targets matched by some criteria.

Target properties can include a default value. This value can provide a general setting that is overridden only on specific targets. For example, you can define a global threshold and allow customization only for a specific target instance where needed.

The list of properties for a target type is inherited from the type's ancestor types, and the type can also define additional properties. You can package target property definitions from the type as well as target-instance property values in automation packs to transfer them between environments. The author has full control of the evolution of the properties of the types, including control of upgrades and other lifecycle management tasks (Figure 1).

Figure 1. Properties of Target Types

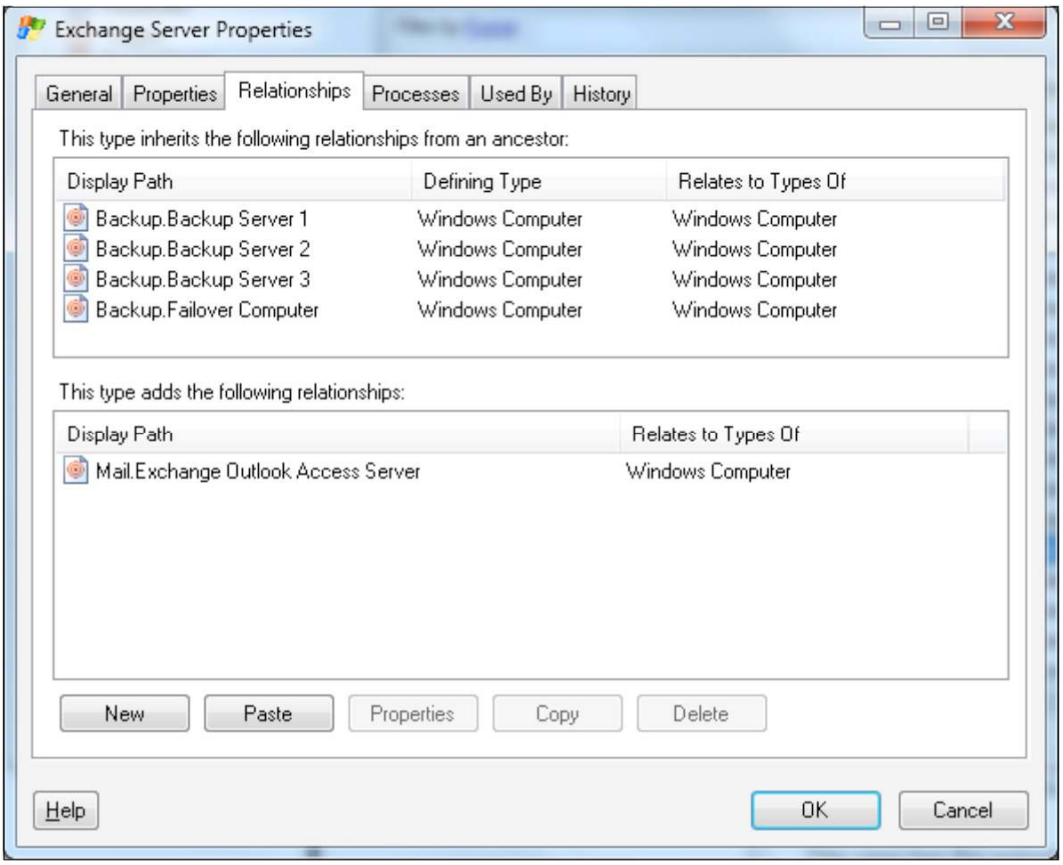


Cisco Process Orchestrator provides a unique implementation of target properties that allows properties to be moved between types. In many cases, property values are even preserved during this restructuring. For example, prior to the use of target types, Cisco Process Orchestrator provided a service target type that allowed arbitrary services to be modeled with a single type. This built-in type still exists and is now called Generic Service. Upgrades will function with no problem. However, this type should be used less now that new types are possible. To represent multiple services using a single type, property name spaces were employed, creating a large number of properties to be overloaded on that single type. Although property name spaces are supported, target types eliminate the need to use name spaces for all properties. Now you can create a new subtype of a Generic Service, move properties in some name space from the Generic Service to the new type, rename the properties to eliminate the name space and simplify the name, and then move the inheritance from Generic Service to some new place in the type structure. The automation pack for the new type can be shipped to a customer, and importing it will upgrade the types and properties to the new structure without loss of the values stored in the properties.

Target Relationships

Relationships enable traversal from one target to another in a workflow. Target types define the relationships. Target instances provide the link to a specific target instance. Like properties, relationships support inheritance. You can model one-to-many relationships with a reference on the “many” pointing to the “one”; then you can use Find Targets to query for targets whose relationship refers to some target (Figure 2).

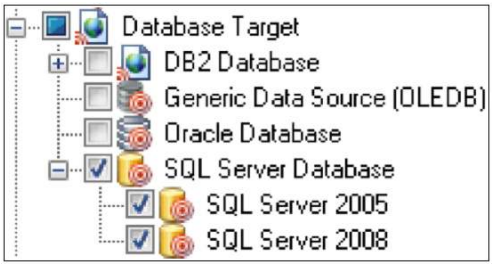
Figure 2. Relationships in a Target



Processes That Act on a Target Type

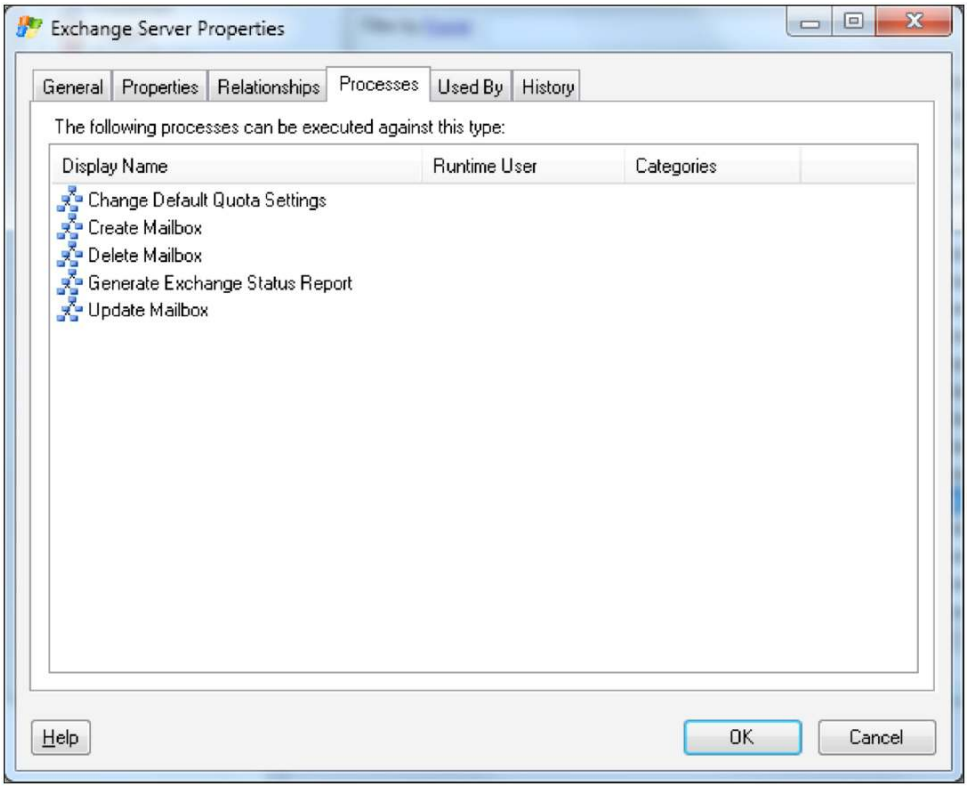
Process definitions control the target types on which they act (Figure 3).

Figure 3. Selection of Target Types in a Process Definition



Select a target type or instance to view the processes that run on that type. This selection supports inheritance. For example, assume that a Microsoft Exchange Server type provides a more specific implementation of a Microsoft Windows computer through inheritance. All processes that can run on a Microsoft Windows computer run using Microsoft Exchange Server; a Microsoft Exchange Server "is a" Microsoft Windows computer (Figure 4).

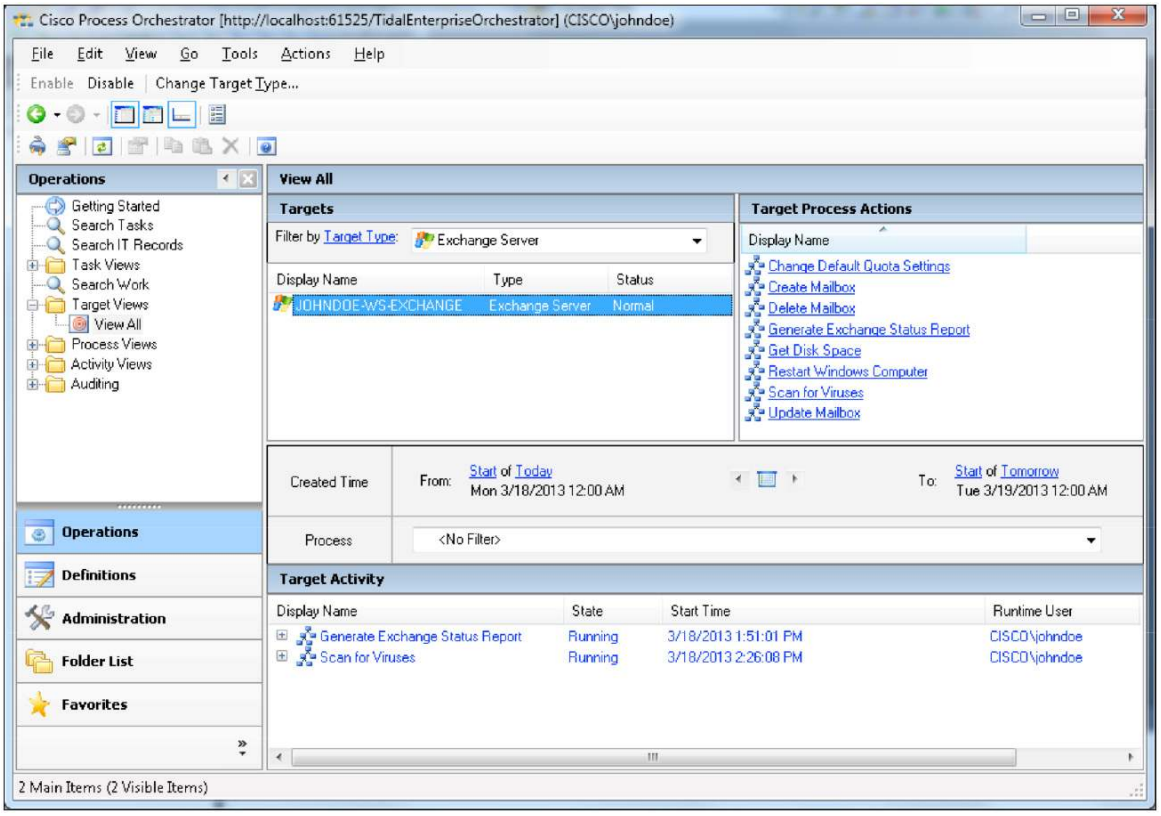
Figure 4. A List of Processes That Act on a Type in a Type Definition



Target Operations Views

Target views of operations allow you to view ongoing automation in the context of the targets or services on which the automation acts. You can filter the list of all targets to focus on a specific type or the name of a target, such as a Cisco Intelligent Automation for Cloud (IAC) service request. Upon target selection, users will see the available processes that act on that target type along with all process activity related to that target (Figure 5).

Figure 5. Target Views in Operations



Interobject Messaging and Policy Provided by Events for a Target Type

Target events complement target types to allow event-based automation for each type of service. When used as triggers, these target events enable policy, allowing you to invoke automation as data is created or changed, according to patterns in the data. Available target events include:

- **Target-created event:** Target properties are available in reference control.
- **Target-changed event:** Both old target and new target properties are available in reference control. Names of changed properties also are available in reference control.
- **Target-deleted event:** Deleted target properties are available in reference control.

Process events allow you to send arbitrary messages from one process to another. These process events can be filtered by target type, providing a type-aware open messaging infrastructure.

Event integration to external systems using AMQP, Simple Network Management Protocol (SNMP), or email is also possible.

Optimization for Cisco Prime Service Catalog

Usability is optimized for interaction with Cisco Prime Service Catalog. You can easily create targets from incoming service requests. Cisco Process Orchestrator helps push service items to the catalog as needed. Authors pick the right moments in a flow to synchronize data. Through the Cisco Prime Service Catalog Adapter, authors can use an active catalog connection to optimize the experience. The Cisco Prime Service Catalog Adapter easily reads, writes, and creates service items and is aware of the Cisco Prime schema. When a service item definition is created in the catalog, it can be integrated rapidly. The property browser exposes the catalog definition to enhance ease of use. Get Service Item properties can be followed by Update Target to save the properties in a single step. You can write directly through the Create Service Item or Update Service Item activities using property references.

Automatic and Consistent API Support

Users and automation pack authors can control the target types published to the northbound web service. For published objects, Cisco Process Orchestrator exposes the type uniquely to the northbound web service. As content teams, services, partners, and customers extend types, these APIs are maintained to provide a consistent format for APIs across all types of objects. However, each type of object is uniquely supported. Authors can choose the properties that are exposed. All exposed properties are automatically accessible through the northbound web service as optional parameters, per type, in the Web Services Description Language (WSDL).

In a typical API, the application provides a static interface to interact with all or some subset of its components. However, when a user is allowed to define an arbitrary set of types, all with custom properties and relationships, a static interface becomes difficult to use. Such an unchanging interface for a service topology might look something like this:

UpdateTarget(string targetTypeId, keyvaluedictionary properties);

Unfortunately, this specification does not provide much direction or insight as to the target types that can be created and the properties that can be created and updated. Because target types and target properties are both user defined, the API must allow for nearly any possible input for the target type as well as any target property name-value pairs. As a result, use of the interface becomes tedious and error prone.

To address this challenge, Cisco Process Orchestrator provides per-type WSDL that defines a specific interface using the then-present properties for the type. The API becomes dynamic, changing according to changes to the type. With the introduction of additional types, additional methods are present, and with the introduction of additional properties, additional parameters for these methods are available. The resulting API is therefore extremely explicit, extremely intuitive, and easy for consumers to use.

As multiple contributors (packaged products, services, partners, and customers) define new types and add new properties to existing types, the API dynamically extends to accommodate the new target types, properties, and relationships. Similarly, as target types or properties are deleted, the API dynamically changes to accommodate the deleted type or property.

For example, assume that a user has defined a new target type called Router. The API will have new methods created for the new target type with parameters that correspond to the user-defined properties. The resulting API for just the Router target type might look like this:

UpdateRouter([optional] string name, [optional] int logLevel);

Notice that Router has two properties, name and logLevel, which are both specified in the type.

Similarly, Cisco Process Orchestrator specifies a per-process WSDL that defines a specific interface to invoke a process using the then-present input and output parameters for the process. The API reflects the current set of process definitions. With the introduction of additional processes, additional methods are created, and with the introduction of additional process parameters, additional API parameters for those methods are created. As with type-specific WDSLs, the resulting API is therefore extremely explicit, extremely intuitive, and easy for consumers to use.

Automation Packs

Like other aspects of configuration, target types can be placed in an automation pack that is version controlled, enabling the partner community to define new types of services and augment those delivered by others.

Service Modeling Examples

This section presents several examples of services created using service-oriented orchestration.

Distributed Application and Product as a Service

Service-oriented orchestration can be used to model a distributed application and manage its provisioning and ongoing operations. To the business, an application and its elements are not important by themselves. The application is important for the service it provides to the business.

For instance, an application may provide customer support, order entry, or partner billing services. A customer support service, for example, can be provided by a distributed application. It consists of a number of web servers behind a load balancer, a database, a cluster of application servers, and a Lightweight Directory Access Protocol (LDAP) repository for user credentials. Each application server ultimately runs on a Cisco Unified Computing System™ (Cisco UCSs) blade, which uses Cisco UCS Manager. Each topology node fits within some network, with each tier separated by firewalls, which require specific firewall configuration, so the application topology has relationships to a network topology. All these elements together form the service topology.

In a platform-as-a-service (PaaS) solution, you need to enable not only provisioning, but also ongoing operations, and eventually deprovisioning of services or applications, whereas for an infrastructure-as-a-service (IaaS) solution, you need to provision individual virtual machines. The leap from what IaaS does to what PaaS does involves treating a service or a collection of virtual machines and their networks as a package. Applications or other services are provisioned and configured on these virtual machines and networks. PaaS involves an ordering experience that allows customers to deploy and use those services.

To this end, service or application blueprints need to be defined that model the services and make them orderable and support provisioning, operations, and deprovisioning of the services. These blueprints should capture what the customer needs to deploy an application. Customers want to receive these blueprints from Cisco or from partners or a community. Customers also need to be able to build these blueprints themselves and move them from development to test to production environments.

Service-oriented orchestration allows you to define, package, ship, and receive these models, as well as to create instances of the service from that model. In this way, deployment of the service can be standardized across development, test, and production systems. For example, a target type can be created for a web server with the specific web server target being an instance created from that web server target type. This instance may have a relationship with the Linux server that hosts the web server.

A level-one helpdesk operator may not know or care what specific database or LDAP instance supports the customer support service or what Cisco UCS blade actually runs a certain instance of the web server. The operator just wants to use the customer support service. For instance, the operator may want to authorize a new user to access the customer support service and then provision whatever the user needs. The operator typically would run an add-user process, acting on the customer support service.

A build process that acts on the customer support service is responsible for provisioning of the whole service. This process may call the build process for each target type in the topology, building the service an element at a time. This process could use files, such as an OVF for a server, to stand up the specific service element, or it might invoke a tool such as Puppet or Chef.

Another example of a process that acts on the service or its elements is site disaster recovery, which may need to make changes to many parts of the distributed customer support application to move it to another site. Such a process needs to act on the customer support service as a whole and use the topology and relationships to traverse to other actions, possibly calling similar processes that act on those elements. For example, a request to back up the overall service might invoke a backup of each server in the topology as well as the database.

Through service-oriented orchestration, processes can act on the customer support service. As necessary, a process can traverse relationships to lower-level infrastructure and tool elements on which it acts. For example, a process that queries capacity might traverse the database target and perform Cisco Process Orchestrator activities against that database.

Cisco TelePresence Service

A Cisco TelePresence® service has several components. Automation can act on these through Secure Shell (SSH), SNMP, and web services. Prior to the use of target types, you had to browse through all processes and possibly filter them by category or automation pack. You would then find and run a process, specifying the terminal, SNMP, or web service target required for the technical method used to achieve the result you wanted. You had to understand the underlying implementation to know what target to provide.

With service-oriented orchestration, an external team provides an automation pack that defines a Cisco TelePresence target type with:

- Relationships to the terminal, SNMP, and web service target types
- Properties such as phone number and escort name
- Process actions for Cisco TelePresence Systems

The automation pack can add properties, such as a location property, to the built-in network device target type.

You install the automation pack and configure Cisco TelePresence service instances by calling a constructor process named Create. The process not only creates the Cisco TelePresence target, but also creates the terminal, SNMP, and web service targets and the relationships that unify them in the model. You then browse the target views or operations views, filter for Cisco TelePresence targets, and select a target. You will see a list of process actions you can take against a Cisco TelePresence target, and you choose an action to run. Internally, the workflow traverses the relationship to find the SSH, etc., target required by the action. You see this and other automation running against the Cisco TelePresence System.

Microsoft Exchange Mailbox

A solution accelerator automation pack defines a Microsoft Exchange Server target type, which inherits from Microsoft Windows Server.

A partner automation pack defines a Microsoft Exchange Mailbox target type. This type includes:

- A Hosted On relationship with a Microsoft Exchange Server
- Properties such as State, Warning Quota, Display Name, and Microsoft Active Directory (AD) Account Name
- Process actions (workflows) such as Create, Delete, and Change Quota Settings

You can then create an instance of the Microsoft Exchange Mailbox target, specifying the Microsoft Exchange Server. A target creation event triggers the Create process action. Alternatively, you can manually run a New Mailbox action from a Microsoft Exchange Server, which embeds the mailbox target creation. The Create process changes the State property to Provisioning, and then the property changes to OK after it successfully completes or Unavailable if it fails. The Create process traverses the Hosted On relationship to the Microsoft Exchange Server to act on it and access all its properties, and it may also traverse Microsoft Exchange Server relationships to additional targets.

You can view the properties of the Microsoft Exchange Mailbox, including the traversal to its Microsoft Exchange Server.

Alignment with IT Standards

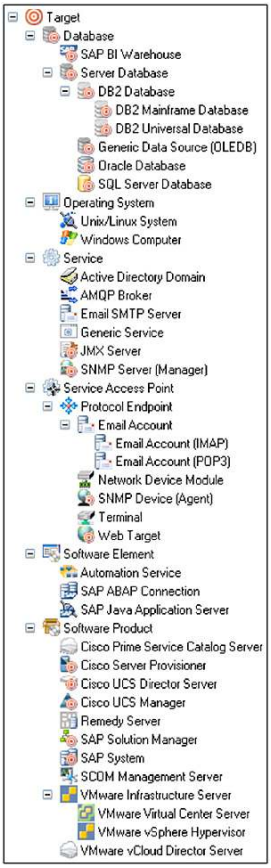
Although service-oriented automation is open, allowing any model, it aligns with IT best practices such as those specified by ITIL. ITIL prescribes a CMDB as a central element that allows high-level services to be decomposed into their supporting lower-level services and ultimately to lower-level systems and devices. In ITIL, each element in the model is called a configuration item because it is a unit of configuration. A CMDB provides a mapping of lower-level elements that support business services. Service-oriented orchestration allows automation to be guided by a model of current and desired IT services with awareness of their relationships and interdependencies. In this way, the principles of service modeling are substantially the same as those for modeling services using ITIL and CMDBs.

However, this implementation bypasses many of the negative features of typical CMDBs. Cisco Process Orchestrator can integrate with a CMDB if it is there. Targets provide a natural synchronization point for exchanging data with a CMDB, and target events allow synchronization to be externalized from other processes. However, service-oriented orchestration does not require a CMDB; you do not need a CMDB for orchestration to function. Often orchestration needs a smaller data set than that in the CMDB, and it is much easier to keep a smaller data set current. For example, resource management can be more efficient when externalized. Moreover, a CMDB typically represents actual elements and not potential elements, and because orchestration is often responsible for provisioning, the orchestrator may include data prior to its use in the CMDB, which it may write to the CMDB as it is instantiated. Even if Cisco Process Orchestrator service models use data in CMDBs, performance is improved because this data remains in the Cisco Process Orchestrator target properties to optimize queries. Cisco Process Orchestrator allows a local representation of only those services that relate to the service that you want to automate, so the implementation is right-sized and lighter than that with a complete CMDB. Updates are more likely to occur in real time and are targeted to meet the needs of automation. A conceptual alignment with ITIL and CMDB principles provides the flexibility to right-size the implementation while allowing unification with CMDBs when needed.

Also, ITIL prescribes critical process records such as alerts, incidents, and change requests that are a focal point for IT operation processes. Cisco Process Orchestrator provides a native representation for these concepts, allowing content to be abstracted from these concepts in the way that they are provided in the organization. For example, content that detects an IT incident need not encode the specific service desk used by a particular customer. Content that enriches incidents with additional diagnostic data can also do so independent of tool selection. Cisco Process Orchestrator alerts, incidents, and change request records allow reusable content independent of tool selection and company-specific ITIL extensions. A separate body of content can integrate orchestrator incidents with a specific service desk such as BMC Remedy. When incidents are created or changed in general automation, Remedy synchronization processes can trigger and push those changes to Remedy. Updates in Remedy, such as incident closure, trigger synchronization processes that update the Cisco Process Orchestrator incident. Processes in the general automation implementation can then proceed from incident resolution. For example, an incident can be verified to prove that the problematic behavior no longer exists. Moreover, ITIL prescribes that these records have a relationship to the element that failed, so that you know the services that may be affected. Cisco Process Orchestrator incidents provide a link to the target defined by the target type for the service; they also provide a secondary configuration item field to link arbitrary external CMDB references when a CMDB is present.

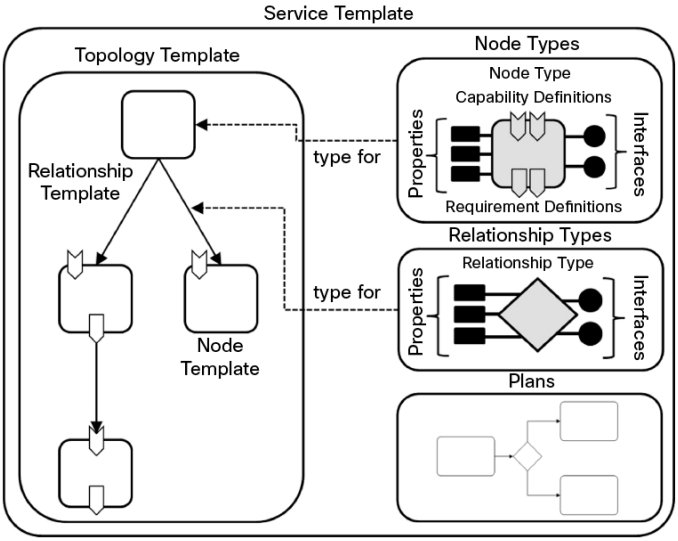
The inheritance hierarchy delivered in the product is based on the DMTF Common Information Model. Although the automation content does not need to use this hierarchy, some implementations will want to align with CIM as a standard. The base product adheres to the standard, so should a customer want to align with CIM, this is possible. All target types are derived from a base target element. The inheritance model is fully extensible (Figure 6).

Figure 6. Packaged Target Types Inheritance Hierarchy: Aligned with the DMTF Common Information Model



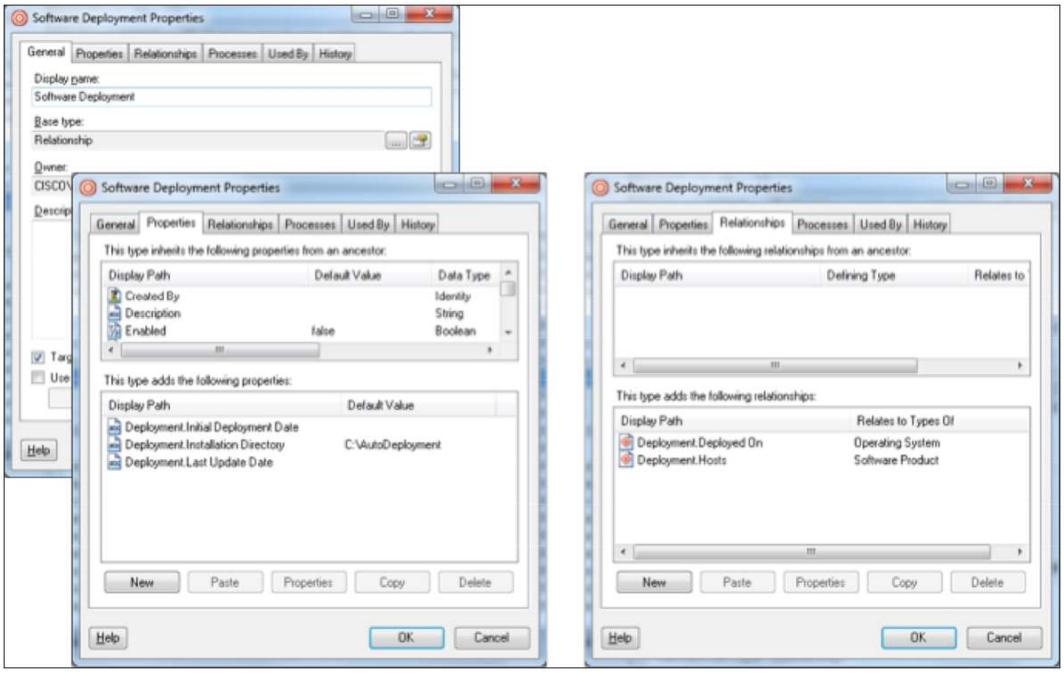
TOSCA standardizes capabilities to define and deploy composite services and applications. TOSCA provides a standard way to define service topologies such as application blueprints. Although Cisco Process Orchestrator does not deliver 100 percent compatibility or interoperability with TOSCA, the implementation does align with TOSCA principles (Figure 7).

Figure 7. Structural Elements of a TOSCA Service Template and Their Relations



- A TOSCA node type defines properties and interfaces. Cisco Process Orchestrator target types address the entire feature set of TOSCA node types and in some cases exceed them. Interfaces are essentially Cisco Process Orchestrator process definitions. In both TOSCA and Cisco Process Orchestrator, types can be derived from other types to inherit their properties, etc. Abstract types are supported.
- Through the use of target types, you can model TOSCA relationship types. By defining a target type for a more complex relationship, you can model many-to-many use cases (Figure 8). Automation uses the Find Targets activity to query for instances of the relationship type whose relationship refers to some target. Additionally, you can add properties for these types that link targets. Because the relationships in this case are targets, processes can act on instances of these relationships. In future releases, Cisco will be enhancing this concept to advance relationships to use Cisco Process Orchestrator's type and property framework with a full relationship type implementation, but you can implement this capability today should your use case demand it.

Figure 8. Target Types Can Be Used to Model Relationship Types



- Plans describe the management aspects of service instances, especially creation and termination. Plans too are delivered through the processes that act on a target type.
- The packaging and shipping of process definitions with target types, properties, etc., is a core value. Cisco Process Orchestrator achieves this benefit through its automation pack capabilities.
- A template allows the service to be ordered as a unit. Ordering is performed in the Cisco Prime Service Catalog, which can combine orders for complex services into a single form. These instances of composite service topologies then flow to the orchestrator through the service request.
- An artifact represents content needed to implement a deployment. Examples include an OVA virtual machine template, an executable file (for example, a script, a Chef recipe, an executable program, or an image), a configuration file, and a data file. Cisco Process Orchestrator can both package files in an automation pack distribution and reference artifacts from a share installed by a higher-level package, and its workflows can push these packages to tools or systems.

Table 1 compares TOSCA features and Cisco Process Orchestrator features and shows the unity of Cisco Process Orchestrator and Cisco Prime Service Catalog in Cisco IAC.

Table 1. TOSCA and Cisco Process Orchestrator Comparison

Feature	TOSCA Specification	Cisco Process Orchestrator	Cisco Process Orchestrator Explanation	Cisco IAC	Cisco IAC Explanation (Includes Cisco Prime Service Catalog)
TOSCA interoperability	Yes	No		No	
Overall conceptual alignment supported in implementation	Yes	Mostly	Node types and relationships are fundamental	Mostly	Node types and relationships are fundamental
Node type	Yes	Full	Target type	Full	Service item definition

Feature	TOSCA Specification	Cisco Process Orchestrator	Cisco Process Orchestrator Explanation	Cisco IAC	Cisco IAC Explanation (Includes Cisco Prime Service Catalog)
Relationship type	Yes	Mostly	Target type implementation; true relationship types planned	Mostly	Service item definition implementation using foreign keys
Plans (workflows)	Yes	Full	Processes and target types	Full	Uses Cisco Process Orchestrator
Derived types	Yes	Full	Target type inheritance	Full	Uses Cisco Process Orchestrator
Abstract types	Yes	Full		Full	Uses Cisco Process Orchestrator
Constraints	Yes	Partial		Partial	Custom catalog ordering model and service items
Requirements and capabilities	Yes	No		Partial	Custom catalog ordering model and service items
Artifact types and templates	Yes	Partial	Can package files in an automation pack	Partial	Uses Cisco Process Orchestrator
Topology template	Yes	Mostly	Requires all the preceding features	Mostly	Service items and Cisco Process Orchestrator
Service template	Yes	Mostly	Requires all the preceding features	Mostly	Service items and Cisco Process Orchestrator
Policy types and templates	Yes	Partial	Uses target types	Partial	Custom catalog ordering model and service items
Archive format	Yes	Mostly	Not all constructs and no Cisco Prime Service Catalog elements, but automation packs are almost identical	Mostly	Uses Cisco Process Orchestrator, plus Cisco Prime Service Catalog allows content export as well
Events, including data-pattern triggers	No	Full	Not in the TOSCA specification, but implemented by Cisco Process Orchestrator	Full	Uses Cisco Process Orchestrator
Automatic API support for all objects	No	Full	Not in the TOSCA specification, but implemented by Cisco Process Orchestrator	Full	Uses Cisco Process Orchestrator
Move properties; properties ship from a variety of contributors	No	Full	Not in the TOSCA specification, but implemented by Cisco Process Orchestrator	Full	Uses Cisco Process Orchestrator

Table 2 summarizes the way that service-oriented automation blends aspects of these industry standards as well as common best practices such as object-oriented programming. Users who are familiar with one of these areas may prefer to map the concepts with which they are familiar to service-oriented orchestration features. As the table shows, similar concepts have very different names across the industry. Cisco Process Orchestrator attempts to use the most intuitive orchestration terms, especially for prior Cisco Process Orchestrator users.

Table 2. Orchestration Terms and Concepts

Cisco Process Orchestrator	Cisco Prime Service Catalog	Object-Oriented Programming	IT Infrastructure Library (ITIL)	Common Information Model (CIM)	Topology and Orchestration Specification for Cloud Applications (TOSCA)
Collection of related targets		Object model	Service Configuration item hierarchy	Model	Service, service template, and topology template
Target type	Service item definition and service standard definition	Class	Configuration item type, or more loosely, asset classes	Class	Node type
Target	Service item and service standard	Instance and object	Configuration item and service	Object derived from managed element	Node
Inheritance	Inheritance, parent, child, ancestor, and descendant	Inheritance, parent, child, ancestor, and descendant Subclass,	Loosely: configuration item categories and classification	Inheritance (subclassing)	Inheritance, parent, child, ancestor, and descendant

Cisco Process Orchestrator	Cisco Prime Service Catalog	Object-Oriented Programming	IT Infrastructure Library (ITIL)	Common Information Model (CIM)	Topology and Orchestration Specification for Cloud Applications (TOSCA)
		superclass, subtype, and supertype			
Property	Attribute	Attribute and field	Configuration item attribute	Property	Property
Future: Relationship type (can be achieved through target types as explained earlier)			Relationship type		Relationship type
Relationship	Relationship	Links and associations Pointers	Configuration item relationship	Relationship and association	Relationship
Process From target type: process action	Action, associated services, and tasks	Method and operation		Method	Plan
Event From process: trigger from event	Event Rules for On Event Some use of the term “trigger”	Event or message		Indication (SNMP, alert, threshold, etc.)	
Target group and target selection	Rule		Selection by configuration item category		
Alert (related to target)			Event (related to configuration item)		
Incident (related to target)			Incident (related to configuration item)		
Change request (related to target)			Change request (related to configuration item)		
Automation pack					Cloud service archive (CSAR)
Files in an automation pack					Artifact
No explicit term The collection of all processes that act on a target type, including their input and output parameters		Class interface		Class interface	Interface
Process target		“This” reference in a method			
“Targets of this type can be created” checkbox on a target type		Abstract class			Abstract type
Can be achieved in processes; see the next section, “Advanced Object-Oriented Principles”		Dynamic binding, overriding, and polymorphism			

Advanced Object-Oriented Programming Principles

As Table 2 shows, service-oriented orchestration supports the major concepts of object-oriented programming. However, Cisco Process Orchestrator supports process-centric (procedural) and service-centric, target-centric,

and object-oriented approaches to orchestration, and it has a large body of existing content that needs to continue to run. Moreover, the design of Cisco Process Orchestrator is intended to allow IT staff who either are nonprogrammers or have little scripting experience to define automation. Therefore, some advanced object-oriented principles are not enforced in the orchestrator itself, but they can be implemented within content for more sophisticated automation authors:

- With encapsulation, the properties of a type should be read and written only by processes of that specific type. Encapsulation can be achieved in an implementation. As a best practice, you should separate all aspects of a target type, so that its properties are manipulated only through its processes (what object-oriented programming calls its class interface). A process should not change properties of another type. With this approach, you can alter the implementation of a target type and its processes without affecting other orchestration elements, avoiding code that is difficult to understand and therefore to fix and modify. Encapsulation is a best practice for service-oriented orchestration but is not mandatory.
- When target instances must be created dynamically, an encapsulation best practice is to use constructor processes to create target instances. This approach can hide the construction of related objects and relationships. For example, creation of a Cisco TelePresence target may also invoke the constructor process for an SSH target and create a relationship from the Cisco TelePresence target to the newly created SSH target, and it may do the same for an SNMP target and a web service target. So the Cisco TelePresence constructor may actually create four objects behind the scenes and relate those objects.
- Cisco Process Orchestrator does not explicitly support override, dynamic binding, or polymorphism. In Cisco Process Orchestrator, process invocation is explicit. However, you can create a process on a base type that is set to be field customizable, so that customers in the field can modify it. This type of process is often called an extension point in Cisco Process Orchestrator. Within this customizable process, users can alter the definition to invoke a descendant type's process to override the default behavior of the base type. This use case can be supported with some planning.
- To help keep Cisco Process Orchestrator usable by nonprogrammers, multiple inheritance is not supported.

Conclusion

Target types and other features supporting service-oriented orchestration enable a significant shift in the approach to automation in which the service or the target, not the process, is the focal point. Automation can define and act on higher-level services that IT provides to the business. Internally, processes follow relationships to the lower-level elements of the IT stack on which they operate. The feature delivers an agile capability to model any service (or any application or process), aligning the user experience more closely with the workflow of the business.

Cisco Process Orchestrator enables the partner community and customers to define new types of services and to extend those shipped by Cisco. New policy capabilities are enabled because automation can be triggered as data is created or changed.

Target types are a focal point for synchronizing data with Cisco Prime Service Catalog service items, CMDBs, discovery, etc. Fluid mechanisms are provided to push and pull information to and from the catalog.

A dynamic API is automatically available for anything modeled or extended. You can create, update, and delete instances, including custom properties, with a per-object WSDL. Additionally, through the API, you can run process actions (methods) that act on the instances.

For More Information

To learn more about Cisco Process Orchestrator, visit http://www.cisco.com/en/US/products/ps11100/index.html.

Find out more about Cisco Prime Service Catalog at http://www.cisco.com/go/service-catalog.

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at www.cisco.com/go/offices.

Printed in USA C11-729698-00 10/13