Introduction

Understanding Service-Oriented Orchestration

Cisco Process Orchestrator 3.5 introduces a new set of features providing Service-Oriented Orchestration that enable a paradigm shift vs. traditional run-book automation and IT process automation. This shift enables automation to align to the high-level services provided by IT, and models how a high level service is supported by a topology of lower level services, systems, and devices.

Planning for services and their desired state are the initial focus of automation design. Focus then moves to defining process actions against these services, with implementation of specific process workflows that traverse these services to act on lower level elements as a final implementation step.This enables a declarative approach to automation, focusing primarily on *what* is desired instead of *how* it is achieved.

Note The capabilities are additive and complimentary to traditional orchestration definitions and approaches, so Cisco Process Orchestrator provides both service-oriented and process-based orchestration. This means that you can still program Process Orchestrator as usual, so you can ease into the new concepts or not use them at all. This guide, however, will demonstrate the service-oriented orchestration approach.

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 achieve automation. The process is the focal point of automation. Processes act on lower-level IT elements such as devices, servers, or specific tools. The set of elements on which automation acts are typically delivered in the product and through adapters connecting to various layers of the IT technology stack. IT, on the other hand, is focused on services providing value to the business, which are much higher up the stack. The inability of RBA/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 creating orchestration active and dynamic, and allow for:

- Defining new, higher-level services in the system, and deploy new services quickly.
- In real-time, after these new types of services have been defined, creating real-time instances of those new services.
- Using events to watch for patterns in these services, enabling policy-driven automation.

Service-Oriented Orchestration combines several industry trends to synthesize a fresh approach to orchestration:

- Service modeling capabilities of a service catalog are now available in the orchestrator layer. Process Orchestrator provides fluid mechanisms to exchange service information with the Cisco Prime Service Catalog, advancing the integration of these systems.
- The feature delivers many of the capabilities of object-oriented design and programming into the RBA / ITPA world. The shift from traditional orchestration to Service-Oriented Orchestration is similar to the shift from procedural to object-oriented programming. Today, virtually all programming is done with object-oriented languages, and object-oriented design has transformed the industry to higher levels of productivity and quality. Service-Oriented Orchestration holds the same promise.
- The IT Infrastructure Library (ITIL) prescribes a service-centric approach for IT. Configuration Management Databases (CMDBs) model IT services and their relationship to other IT assets. Service-Oriented Automation allows automation to be driven through a model of current and potential IT services, aware of their relationships and interdependencies. In this aspect, the principles of service modeling in Process Orchestrator are essentially the same as modeling services within ITIL and CMDBs. While Process Orchestrator can integrate with an available CMDB, there is no requirement to have a CMDB to enable orchestration.
- The feature aligns to industry standards like the DMTF Common Information Model (CIM) and the Topology and Orchestration Specification for Cloud Applications (TOSCA).
- Model-based automation is becoming popular via script-based tools, especially in the configuration management space. Process Orchestrator combines the capability to model services with the openness to integrate with these tools to leverage their strengths. Moreover, the feature allows model-driven orchestration atop legacy tools to bring the full power of model-driven approaches to integrate with other IT tools.

Key Benefits

Service-Oriented Orchestration allows automation to focus on higher level IT services, possibly specific to the customer's business and unknown by Cisco when the product is built. User interaction shifts to services on which to act and what you can do to them. The approach to defining automation shifts from having to first decompose automation into a sequence of *processes*, to decomposing high-level *services* into their components, and then defining actions that are possible on each of the component services.

This inversion in approach is simple, yet powerful. Cisco Services, partners, and customers can model services and extend others' services without coding. Extensions and automation can be packaged, shipped to customers or moved from development to test to production, versioned, and upgraded.

The approach delivers several key benefits:

- Greater ease of use in combining Process Orchestrator with Prime Service Catalog.

  The Prime Service Catalog adapter simplifies the exchange of service requests and service items to and from the orchestrator. It is easy to create targets from incoming service requests. Process Orchestrator helps push service items back to the catalog as needed by allowing users to pick the right moments in a flow to synchronize data. Using active catalog connection in the Prime Service Catalog adapters optimize user experience. The Prime Service Catalog adapter easily reads, writes, or creates service items, and is CP schema-aware. When a service item definition is created in the catalog, the desire is to set up integration 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. One can write directly through the Create Service Item or Update Service Item activities using property references.
- It allows simpler, more readable workflows.

- Data defines desired services, and the service instance drives automation to achieve the desired state. This separates the desired state from the implementation (the “what” from the “how”).
- It acts on the higher-level service rather than its technology elements. Services can span tools, and workflows can navigate service topologies to lower-level elements on which they act.
- It provides operational views of automation by service.
- It lets you monitor the environment vs. the service definition and bring them in line with policy.
- It provides federated storage, including push objects and relationships to and from Service Catalog, CMDB, and Service Assurance tools when needed.
- Automation becomes easier to extend and customize.

Process Orchestrator Supports Service-Oriented Orchestration

Several features in Process Orchestrator combine to bring these capabilities:

- Service instances are **targets**. A **target type** allows you to define a new service; all new targets are created based on a target type. Target types can be based on the DMTF Common Information Model, but this is not required.

  Target types are service definitions that allow Process Orchestrator to deliver service-oriented automation, where the *service*, not the process, is the focal point. In service-oriented automation, the *content*, not the Process Orchestrator platform, defines the solution models; the platform is open to any model you want to produce.

  Some target types can be ‘abstract’, meaning they cannot be directly instantiated into targets but are only available for inheritance by other target types. In Process Orchestrator, these target types are marked as either ‘creatable’ or ‘not creatable’.

  Target types support inheritance, which allows you to extend a general type for a specialized need. In Process Orchestrator, you can select a ‘base target type’ for a target type that specifies that the target type inherits from the base target type.

  Target types have an extensible list of **properties** including field-customizable default values.

  In these properties, data can be stored that might come from Process Orchestrator in terms of defining an order or a desired state, or might include data that is collected from the environment. For example, periodic network device discovery can store information such as the operating system, version, which line cards are installed, and so on. Network automation can use that information in its workflows.

- **Relationships** allow modeling of topologies of lower level services assembled to offer higher-level services; that is, relationships allow lower-level objects to be bound together to achieve a larger whole. Automation can navigate from a higher-level object, such as a service, to lower-level objects against which they can perform automation.
  - Relationships allow you to navigate from one object to another in a workflow.
  - Target types define the property that models the relationship.
  - Target instances provide the link to a specific instance.
  - Relationships support inheritance.
  - Relationships can be set in the northbound web service like any property.
- **Processes** provide actions that can be executed against targets. Process Orchestrator allows you to view a target to see what actions are possible, or what automation is being done against those targets.

  The target types that a process can accept define the services on which a process can act.

- Targets and target types have **events**. These can be internal process events or the open Advanced Message Queuing Protocol (AMQP) protocol. Triggering processes in response to patterns in underlying processes provides policy.
- **Extensible by services, partners, customers**. The content, not the platform, defines service models. The platform is open to model any IT service topology within automation content. For example, some types, actions, properties, etc. may come from the 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 control and lifecycle of the elements they deliver so they can deliver upgrades.
- A **consistent API**, even if elements come from different authors. Users and automation pack authors (see Automation Packs) can control which target types are published to the NorthBound Web Service (NBWS; see the *Northbound Web Services Guide*). For published objects, Process Orchestrator exposes the type uniquely in the NBWS. As content teams, services, partners, or 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 also configure which properties are exposed in the NBWS as optional parameters in the per-type WSDL.

Aligned to IT Standards

While Service-Oriented Automation is open, allowing any model, it aligns to IT best practices such as the IT Infrastructure Library (ITIL). ITIL prescribes a configuration management database (CMDB) as a central tenet which allows high level services to be decomposed into their supporting lower level services and ultimately to the lower level systems and devices. In ITIL, each element in the model is called a Configuration Item since it is a unit of configuration. This provides a mapping of how lower level elements support business services. Service-Oriented Orchestration allows automation to be driven through a model of current and desired IT services, aware of their relationships and interdependencies. In this aspect, the principles of service modeling are substantially the same as modeling services within ITIL and CMDBs.

However, the implementation bypasses many of the negatives and weight of typical CMDB implementations. Process Orchestrator can integrate with a CMDB if it is there. Targets provide a natural synchronization point to exchange data with a CMDB, and target events allow that synchronization to be externalized from other processes. However, Service-Oriented Orchestration does not require a CMDB; one does not have to have a CMDB to enable orchestration to function. Often orchestration needs a reduced model vs. what is in the CMDB, and it is therefore much easier to keep the 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 since orchestration is often responsible for provisioning, the orchestrator may have data prior to its life in the CMDB, which it might write to the CMDB as it is instantiated. Even if Process Orchestrator service models leverage data in CMDBs, performance is improved by keeping this data in Process Orchestrator target properties to optimize queries. Process Orchestrator allows a local representation of only those services that relate to what you want to automate, so that the implementation is right-sized and lighter than complete CMDBs. Updates are more real time and targeted to the need of automation. A conceptual alignment to ITIL and CMDB principles creates the flexibility to right size the implementation and unify with CMDBs when needed.

Also, ITIL prescribes key process records such as alerts, incidents, and change requests that drive the IT operations process. Process Orchestrator provides a native representation for these concepts. This allows content to be abstracted from how these concepts are provided in a company. For example, content that detects an IT incident does not have to encode the specific service desk used by a particular customer.

Content that enriches incidents with further diagnostic data can also do so independent of tool selection. 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 like BMC Remedy. Processes can trigger when incidents are created or changed in general automation and push those changes to Remedy. Updates within Remedy such as incident closure trigger synchronization processes which update the Process Orchestrator incident. Processes in general automation can then proceed from incident resolution. For example, the incident can be verified to prove that the problematic behavior no longer manifests. Moreover, ITIL prescribes that these records have a relationship to the element which failed so one knows what services may be impacted. Process Orchestrator incidents provide a link to:

- The target defined by the target type for the service
- A secondary configuration item field to link arbitrary external CMDB references when a CMDB is present

Related Service-Oriented Orchestration Terms and Concepts

The following table summarizes how Service-Oriented Automation blends aspects of these industry standards as well as common best practices like object-oriented programming. Users who are familiar with one of these domains might find it easiest to map the concepts with which they are familiar to Service-Oriented Orchestration features. As you can see, the feature synthesizes similar concepts that have very disparate terms across the industry. Process Orchestrator attempts to provide the most intuitive terms within orchestration, and especially with prior Process Orchestrator users.

Table 1-1 Service-Oriented Orchestration Terms and Concepts Comparison

Object-Oriented Programming	IT Infrastructure Library (ITIL)	Cisco Process Orchestrator	Cisco Prime Service Catalog	Common Information Model (CIM)	Topology and Orchestration Specification for Cloud Applications (TOSCA)
Object Model	Service Configuration Item Hierarchy	Collection of related targets		Model	Service, Service Template, Topology Template
Class	Configuration Item type, or more loosely asset classes	Target Type	Service Item Definition, Service Standard Definition	Class	Node Type
Instance, object	Configuration Item, Service	Target	Service Item, Service Standard	Object derived from Managed Element	Node
Inheritance, Parent / child / ancestor / descendant Subclass / superclass, subtype / super-type	(loosely) Configuration Item categories and classification	Inheritance	Inheritance, Parent / child / ancestor / descendant	Inheritance (subclassing)	Inheritance, Parent / child / ancestor / descendant

Table 1-1 Service-Oriented Orchestration Terms and Concepts Comparison (continued)

Object-Oriented Programming	IT Infrastructure Library (ITIL)	Cisco Process Orchestrator	Cisco Prime Service Catalog	Common Information Model (CIM)	Topology and Orchestration Specification for Cloud Applications (TOSCA)
Attribute, field	Configuration Item attribute	Property	Attribute	Property	Property
	Relationship type	Future: Relationship Type (achievable through Target Types - see above)			Relationship type
Links and associations Pointers	Configuration Item relationship	Relationship	Relationship	Relationship, association	Relationship
Method, operation		Process From Target Type: Process Action	Action, Associated Services, Tasks	Method	Plan
Event or Message		Event From Process: Trigger from Event	Event Rules for “On Event” Some use of the term “Trigger”	Indication (stop, Alert, Threshold, and others)	
	Selection by Configuration Item Category	Target Group, Target Selection	... Rule		
	Event (related to Configuration Item)	Alert (Related to Target)			
	Incident (related to configuration item)	Incident (Related to Target)			
Change Request (related to configuration item)		Change Request (Related to Target)			
		Automation Pack			Cloud Service Archive (CSAR)
		Files in an Automation Pack			Artifact
Class Interface		No explicit term The collection of all processes that act on a Target Type, including their input and output parameters		Class Interface	Interface

Table 1-1 Service-Oriented Orchestration Terms and Concepts Comparison (continued)

Object-Oriented Programming	IT Infrastructure Library (ITIL)	Cisco Process Orchestrator	Cisco Prime Service Catalog	Common Information Model (CIM)	Topology and Orchestration Specification for Cloud Applications (TOSCA)
"This" reference in a method		Process Target			
Abstract class		"Targets of this type can be created" check box on a Target Type			Abstract type
Dynamic binding, overriding, polymorphism		Achievable in processes			

Service Definition Examples

The following sections provide some service definition examples.

A Distributed Application and Product as a Service

An example of the usefulness of Service-Oriented Orchestration is the ability to model a distributed application and provide 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 example, an application may provide customer support, order entry, or partner billing. Let's pick one of these, say customer support. The customer support service in this example is provided by a distributed application that consists of a number of web servers behind a load balancer, a database, a cluster of application servers, and an LDAP repository for user credentials. Each application server ultimately runs on a UCS blade, which is in turn related to a UCS Manager. Moreover, each of these topology nodes fit within some network, with each tier separated by firewalls, and these nodes require specific firewall configuration, so the application topology has relationships to a network topology. All of this combines to form the service topology.

In a Platform as a Service solution, the customer must enable not only provisioning but ongoing operations and eventually deprovisioning of services or applications, as opposed to IaaS where individual virtual machines are provisioned. The leap 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 VMs and networks. It involves an ordering experience that allows customers to deploy and use those services.

To this end, there is a need to define service or application blueprints so they can model their services and make them orderable and support provisioning, operations, and de-provisioning of those services. These blueprints capture what the customer needs to deploy some application. Customers want to receive these blueprints from Cisco, or possibly from partners or a community. They also need to be able to build these blueprints themselves and move them from development to test to production.

Service-Oriented Orchestration allows one to define, package, ship, and receive these models, as well as to create instances of the service from that model. In this way, the company can drive a standardized deployment of the service across development, test, and production systems. For example, a target type could be created for a web server, while the specific web server target is an instance created from that web server target type. This might have a relationship to the Linux server that hosts the web server.

A level 1 help desk operator might not know or care what specific database or LDAP instance supports the customer support service, or what UCS blade actually runs a certain instance of the web server. They want to do things against the customer support service. Say they want to authorize a new user to access the customer support service and provision whatever they need. They would generally 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 of the service. This process might call the Build process for each target type in the topology, building the service up an element at a time. This process might use files, like an OVF for a server, to stand up the specific service element, or might invoke a tool such as Puppet or Chef.

Other examples of processes that act on the service or its elements are things like site disaster recovery, that may need to make changes to many parts of the customer support distributed application to move it to another site. These processes need to act on the customer support service as a whole, but leverage 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, the process can traverse relationships to lower level infrastructure and tool elements on which it takes action. For example, a process that queries capacity might traverse to the database target, and perform Process Orchestrator activities against that database.

A Cisco TelePresence Service

A TelePresence system has a number of components that automation can act on through SSH, stop, or a web service. Prior to Process Orchestrator 3.0, to act on a service, one was required to browse through all of the processes, filter by category or automation pack (see Automation Packs), then run a process and specify the terminal, stop, or web service target. One had to understand the underlying implementation to know what target to provide.

Now, using Service-Oriented Orchestration:

1. An external team provides an automation pack that defines a TelePresence target type, with:
   - Relationships defined for terminal, stop, and web service target types
   - Properties such as a phone number or escort name
   - Process actions for TelePresence systems
2. The automation pack can add properties such as a location property to the built-in network device target type.
3. The customer installs the automation pack and configures TelePresence service instances by calling a constructor process called **Create**.
4. The process not only creates the TelePresence target, but also creates the terminal, stop, and web service targets as well as the relationships that unify them into the model.
5. The end user can browse the target views or operations views and filter for targets with the TelePresence type. When they select a type, they can see:
   - All of the available user-startable actions (processes).

  - All automation running against the TelePresence system. These results are filterable by a time range or by a specific process.

6. When the end user runs a process action, internally the workflow traverses the relationship to find the SSH, etc. target required by the action. The user sees this and other automation running against the TelePresence system.

Process Orchestrator System Components

The topics in the following sections describe the major architectural elements of the Process Orchestrator.

Process Orchestrator Console

The Process Orchestrator console is a Windows form-based UI that is intended for advanced users that will be defining processes, target types, activities, and so on.

The console contains the following workspaces, each of which contains a group of objects that perform specific actions within the application.

- Operations—Use the Operations workspace to monitor the orchestration that is executing or is scheduled to execute on various targets. You can also use this workspace to monitor the processes that are scheduled to execute, view processes that are currently running, and verify that processes have successfully completed.
- Definitions—Use the Definitions workspace to view and modify processes and other related configurations that are used in executing automation.
- Administration—Use the Administration workspace to perform administrative actions such as editing system-wide settings, managing security roles, configuring adapters and automation packs, and tuning database grooming and other settings.

Web Console

The Process Orchestrator Web Console provides a web-based UI that is intended for occasional users. The vision is that while a broad set of users may be defining processes, pervasively everyone in IT and even non-IT individuals may need to interact with those processes. The web console focuses on this second group. It does not allow users to define processes but does allow them to start processes, monitor processes they started, or interact with human steps in processes called *tasks* (such as approvals). This interface is purposefully simplified to minimize the learning curve of occasional users.

Note The Web Console is typically not used when Process Orchestrator is used in conjunction with Cisco Prime Service Catalog. In these cases, end-user interaction occurs through the catalog rather than Process Orchestrator interfaces. The Web Console is typically only used in cases when Process Orchestrator is used independently.

The web console runs on a Microsoft IIS web server. Process Orchestrator environments can span Process Orchestrator servers, so if a deployment scenario requires breaking up automation for security, organizational, geographic, or purpose (such as development or test), the Web Console can provide a single point of access for occasional users.

Related Topics

Chapter 9, "Performing Basic Console Tasks"

Process Orchestrator Server (Process Engine)

At the core is the Process Orchestrator Server, which executes automated processes. This component is responsible for:

- Creating instances of running processes
- Orchestrating the execution of these processes
- Configuring multiple servers to work together for load balancing and in an active-active configuration to achieve high availability.

Northbound Web Service Interface

Process Orchestrator provides a web service interface that allows external systems to invoke Process Orchestrator processes, check their results, and so on. Use the northbound web service interface when you:

- Have a need to integrate with or communicate between multiple Process Orchestrator environments
- Have your own support portal
- Use a service catalog
- Leverage Process Orchestrator underneath some other master orchestrator

For more information about the northbound web service interface, see the *Cisco Process Orchestrator Northbound Web Services Guide*.

REST Services Guide

Process Orchestrator provides a web service interface that allows external systems to invoke Process Orchestrator processes, check their results, and so on. Use the REST services guide when you:

- Have a need to integrate with or communicate between multiple Process Orchestrator environments
- Have your own support portal
- Use a service catalog
- Leverage Process Orchestrator underneath some other master orchestrator

For more information about the REST services guide, see the *Cisco Process Orchestrator REST Services Guide*.

Command Line Interface

The Process Orchestrator command line interface is based on Windows PowerShell. PowerShell is the emerging standard for CLI interfaces on the Microsoft platform. The CLI provides basic operations that are needed for integration, including the ability to:

- Export or import automation packs

- Start a process with input variables
- Check the status of a process
- Enable and disable processes and targets
- **Import automation pack patches**
- **Export/Import automation pack customizations**

Databases

Process Database (Configuration & Audit)

This database is closely associated with a Process Orchestrator high availability environment (that includes one or more servers) and stores process definitions and other configuration information. In this environment, the Process Orchestrator Server (Process Engine):

1. Loads process definitions and other configuration information at startup. Then, as appropriate, it starts instances of processes.
2. Stores the state of running process instances into the Process Database. This persistence allows resiliency across server restarts and supports load balancing and high availability in an multi-server environment.
3. Stores auditing data in the Process Database to record change as users interact with the system, (such as to make a configuration change or invoke a process manually).

Reporting Database (Data Warehouse)

This second database stores data needed for longer-term retention and storage. This can include a copy of auditing data that can groom more slowly than the Process database, as well as summary information for each process that ran. This data is organized for reporting, with flat relational tables, and can be summarized. Automation content can also add data, such as performance measurements, to the reporting database.

Related Topics

Reporting

Reporting

Process Orchestrator supports two reporting technologies:

- SAP BusinessObjects
- SQL Server Reporting Services

Report definitions are provided in the Process Orchestrator product and in some Cisco automation packs, and are imported into the chosen reporting platform.

Related Topics

Managing the Report Database

Role-Based Access Control (RBAC)

Windows performs authentication of the user connecting to Process Orchestrator against Active Directory. The authentication is done prior to connection to the Process Orchestrator server, and the Process Orchestrator server simply relies on the results of that authentication. No special end-user account management is necessary for Process Orchestrator.

In Process Orchestrator, authorization is performed using a Role-Based Access Control System. Roles are a collection of permissions. Users are assigned to roles that give them the ability to perform the actions allowed by the role.

Typically, roles are defined according to a standardized job function within IT. Examples might include Level 1 Helpdesk, Level 2 Helpdesk, Cloud Technical Administrator, Network Configuration, SAP Basis Expert, and so on. Security groups already in the directory for the users in these job functions are then typically assigned to the roles.

Adapters

Adapters are one of the extensibility mechanisms in the Process Orchestrator platform, and are only written by members of the Process Orchestrator product development team. The development team uses adapters to extend Process Orchestrator functionality to integrate with devices, environments, applications, or tools without undergoing core modification.

Related Topics

- Configuring Security

Automation Packs

Automation packs are the primary means of extending integrations in the field. Automation packs are prepackaged collections of process definitions, target types, variables, categories, targets, target groups, and other configurations needed to define a set of automated IT processes. Automation packs allow:

- Cisco or its partners to ship best practice automation to customers.
- Customers to get productive quickly, and to shift automation across systems, such as from a development system to a test system to production.
- Customers to export their own processes and share with each other in the community.
- The backup of process definitions and other product configuration.

Automation packs:

- Can be exported from one Process Orchestrator system and imported into another.
- Can ship separately from product releases.
- Are versioned and dependencies are recorded. Versions and dependencies are checked during import to ensure that the system will work after the import. For example, if your personal automation pack calls a process in “Sample Document A” automation pack v1.3, then you cannot import your automation pack into a system that has installed “Sample Document A” v1.2.

The Core automation pack is provided by Cisco Process Orchestrator and is a required prerequisite for all other automation packs included in the Cisco Process Orchestrator installation.

Related Topics

- Managing Automation Packs

Advanced Message Queuing Protocol (AMQP)

The Advanced Message Queuing Protocol (AMQP) is an open standard application layer protocol for message-oriented middleware. It delivers message orientation, queuing, routing (including point-to-point and publish-and-subscribe), reliability and security. AMQP came out of the financial industry, and is proven highly scalable in demanding environments (see http://www.amqp.org/).

AMQP is the emerging standard for messaging and events in the cloud. For example:

- vCloud Director can publish vCloud Messages, also known as “blocking tasks,” “notifications,” or “call-outs,” related to different provisioning. An orchestrator can not only receive these events, but can also respond to them to delay execution.
- AMQP is supported with vCloud Orchestrator.
- OpenStack selected AMQP as the messaging technology for OpenStack.

AMQP is integrated with more than 70 developer platforms, providing a nice framework for event-oriented integrations.

AMQP enables event-driven capabilities of Enterprise Service Bus-style designs. It enables queuing, so automation can fetch messages when there is capacity. You can use asynchronous methods servicing requests as capacity allows when possible, and reserve real-time, synchronous methods only when absolutely needed. AMQP’s open platform is a natural choice for event and message design patterns

Process Orchestrator can trigger processes in response to messages placed on AMQP queues/exchanges. Processes can also read messages from queues/exchanges one at a time if they want to respond to messages one at a time rather than in parallel, to operate more as a queue. Processes can submit messages to queues/exchanges.

JMS is another possible integration enabled through AMQP. AMQP has providers to place JMS messages on queues/exchanges.

Related Topics

Integrating Using the Advanced Message Queuing Protocol (AMQP)

AMQP Exchange Types

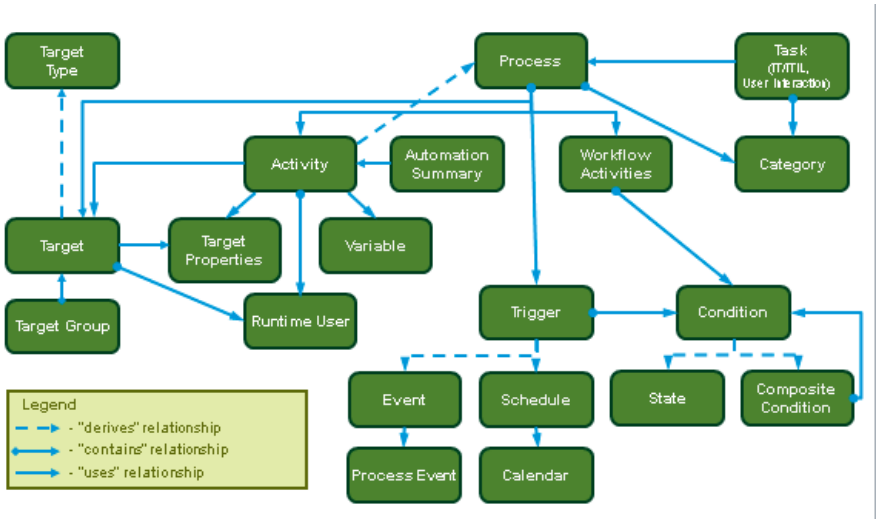
The exchange type determines how AMQP processes and routes messages. There are several types of exchanges.

Exchange Type	Description
Default	The default exchange is a direct exchange with no name (empty string) pre-declared by the broker. It has one special property that makes it very useful for simple applications: every queue that is created is automatically bound to it with a routing key which is the same as the queue name. For example, when you declare a queue with the name of "search-indexing-online", the AMQP broker will bind it to the default exchange using "search-indexing-online" as the routing key. Therefore, a message published to the default exchange with the routing key "search-indexing-online" will be routed to the queue "search-indexing-online". In other words, the default exchange makes it seem like it is possible to deliver messages directly to queues, even though that is not technically what is happening.
Direct	Delivers messages to queues based on the message routing key. A direct exchange is ideal for the unicast routing of messages (although they can be used for multicast routing as well).
Fanout	Routes messages to all of the queues that are bound to it and the routing key is ignored. If N queues are bound to a fanout exchange, when a new message is published to that exchange a copy of the message is delivered to all N queues. Fanout exchanges are ideal for the broadcast routing of messages.
Topic	Routes messages to one or many queues based on matching between a message routing key and the pattern that was used to bind a queue to an exchange. The topic exchange type is often used to implement various publish/subscribe pattern variations. Topic exchanges are commonly used for the multicast routing of messages.
Headers	Routes messages based on multiple attributes that are more easily expressed as message headers than a routing key. Headers exchanges ignore the routing key attribute. Instead, the attributes used for routing are taken from the headers attribute. A message is considered matching if the value of the header equals the value specified upon binding.

Process Orchestrator System Elements

The following diagram shows the major functional elements of the Process Orchestrator system. These elements are discussed in the sections that follow.

Figure 1-1 Process Orchestrator Functional Model



Activities

Activities are the steps in a process. They are customized to perform integration with some environment. For example, an stop trap send activity would have a very different UI and set of properties from an SAP ABAP call, and would have very different runtime characteristics as well.

Activities can be provided by adapters (binary components in Cisco Process Orchestrator) or by automation packs. Therefore both adapters and automation packs can contribute to a particular integration.

Workflow activities provide the logic or flow aspects of the process workflow. Workflow activities are exposed in the Logic tab of the toolbox in the Process Editor.

Related Topics

- Advanced Authoring Concepts
- Adapters
- Automation Packs
- Adding Logic Components to a Process

Automation Summaries

Technically, automation summaries are XML files that are published on a share or web server. The path or URL is then available to be linked from anywhere. This architecture makes them available for integration into virtually any context.

Automation Summaries are a key integration tool. For example, they can be placed in incidents in the Service Desk to reveal the analysis of the automation that was performed, or they can show how an incident was resolved.

Conditions

Many workflow logic elements perform tests to control execution. Conditions implement these tests. For example, a Condition Branch can split execution to take one path if a condition exists, and another if it does not. A While Block can iterate execution while a condition exists.

Conditions can also be placed on a process trigger, allowing control of situations in which the process can run. For example, a scenario might require the process to run during two different time ranges, which would require two triggers: one trigger with an 8am-5pm condition and a totally different trigger with a 5pm-8am condition.

There are two basic types of conditions:

- State conditions

  State conditions evaluate some state. The Process Orchestrator provides a number of state conditions that can be perceived as a part of the base product, but technically are part of the adapter. For example, it provides a test to see if a variable has some value.

- Composite conditions

  A composite condition builds a compound condition from individual conditions. It allows combining conditions with AND logic, where all of the conditions must be TRUE for the composite condition to be TRUE, or OR logic, where any of the conditions can be TRUE for the composite condition to return TRUE.

  Compound conditions can be used in other compound conditions to produce complex logic, such as ((X AND Y) OR (A AND B)).

Processes

Use Process Orchestrator to automate an IT processes by defining a process, then running instances of the defined process.

An element of the process is the process workflow. The workflow defines the automation steps (activities), the logic or flow between these steps, and how to flow data from one step to the next.

The engine manages the state and lifecycle of a process, bringing it into existence, running its steps, and finally terminating it. During and (by default) after process execution, the engine retains information so that operators can view the status of their running processes.

Related Topics

Creating a Basic Process

Runtime Users

A runtime user record stores information about the user security context and passes this information to the adapters for activity execution, event monitoring, and some target operations (such as availability monitoring and discovery). Runtime user instances can be shared across targets and processes. For example, if a single set of credentials can be used to access a set of network devices, only one runtime

user instance must be created. When it is time to change the credentials, users can go to the runtime users list and edit the single instance to change the credentials. This greatly reduces the configuration load when credentials tend to change often in some environments.

Note Runtime user credentials can be used in a process, but no process can retrieve credentials. If your process must access credentials, use hidden string variables.

The runtime user concept allows the product to implement delegation. For example:

1. An IT help desk operator comes to Process Orchestrator to run a process.
2. This operator is presented with a list of processes that Process Orchestrator's role-based access control allows them to run. These processes might include activities that require a level of security permission that the operator does not natively have.
3. The operator can perform actions as a part of the established process that are not possible for them to perform manually.

This concept can also be leveraged to reveal where operators make changes outside of a process. By examining auditing logs such as Windows logs for things being done under the operator's credentials rather than the Process Orchestrator runtime user credentials, it is possible to determine how the operator is doing things outside of process and determine how to close things down. So a side effect of Process Orchestrator automation is that customers might be able to tighten security in their environment.

Targets

Targets are *instances* created from a target type. For example:

- A terminal target allows SSH or telnet to some specific network device.
- A UCS Manager target allows connection to a specific UCS Manager in charge of a UCS system.
- A database target allows connection to specific supported databases.

A process or activity executes an action within some environment. The specifics of the definition of the connection to the environment are encompassed in the target. For example, a target for:

- An SSH command might be a specific UNIX system or network device.
- A database query might be a specific database
- An SAP ABAP activity might be a specific SAP system.

Processes can restrict the type of target which they can accept. For example, it is not appropriate to run a process to change a network device's configuration against a Windows computer.

A process workflow acts on a target. This allows the process workflow to perform actions against an external tool or environment. Although a default target instance can be specified directly by a process, more commonly it is associated to the process at run time.

Activities, which are the steps in a process (see Activities), can also specify a target. Typically, activities default to use the process target, but a step in a workflow might need to happen against a different system than the process target. For example, a process overall might deal with a network device, but a step in the process might need to update a database. Often workflows can determine the target on which an activity may act by using a relationship to the target for the process, or doing a query on some data such as a name to find a matching target instance.

Related Topics

- Defining a Target
- Creating Triggers

Target Groups

Target groups are collections of targets. Often automation might need to run against all machines in a collection, or against one of the machines in a collection. Target groups provide this functionality.

Adapters can provide target groups to leverage grouping definitions where they exist. For example:

- Active Directory OU—Customers are frustrated when they must recode their grouping information into yet another product. The Active Directory adapter seeks grouping information where it is defined. For example, an Active Directory target group looks up computers in some organizational unit (OU) in the directory.
- Target Type group—Using queries of their attributes, the Process Orchestrator provides type-based groupings of targets. For example, use a target group to group all targets of a given type, or to perform additional filtering to pattern match against a field in the target definition.
- Virtual group—A target group can be a virtual group. Use a virtual group to specify an explicit list of targets, providing the capability to manually select targets and establish group membership. A virtual group can also allow the inclusion of other target groups so that group membership can be defined hierarchically. For example, a virtual group called “Production switches” might include all members of the “Houston data center switches” group as well as all members of the “London data center switches” group, but not members of the “Engineering lab switches.”

A default target for a process can specify a target group along with a target selection algorithm:

- Typically, a target selection algorithm chooses one or more members of a target group to specify the target instances on which the process will act.
- At runtime the target selection algorithm resolves the then-present members of the target group and selects a target.
- When a user, CLI, or northbound API call interactively runs a process ad-hoc (on demand), the user can accept the default target specified in the process or override it with a specific target.
- Where a target selection algorithm resolves to multiple target instances, the engine spins up separate process instances for each target. Thus at runtime, a process instance has a specific target on which it will act, against which the process workflow encodes actions.

Related Topics

Defining a Target

Target Types

Target types provide a way to define a service or other IT element that is not represented by any target type provided by an adapter. A target type can:

- Extend an existing adapter-provided target type
- Extend another target type
- Define a completely new target type

All new targets are created based upon a target type. Some target types are ‘abstract’, meaning they cannot be directly instantiated into targets but are only available for inheritance by other target types. In Process Orchestrator, these target types are marked as either ‘creatable’ or ‘not creatable’.

A target type:

- Exposes (and inherits from its ancestor target types) properties and inter-target named relationships that can be read and set either manually or by an Update Target activity.
- Defines property default values.

Tasks

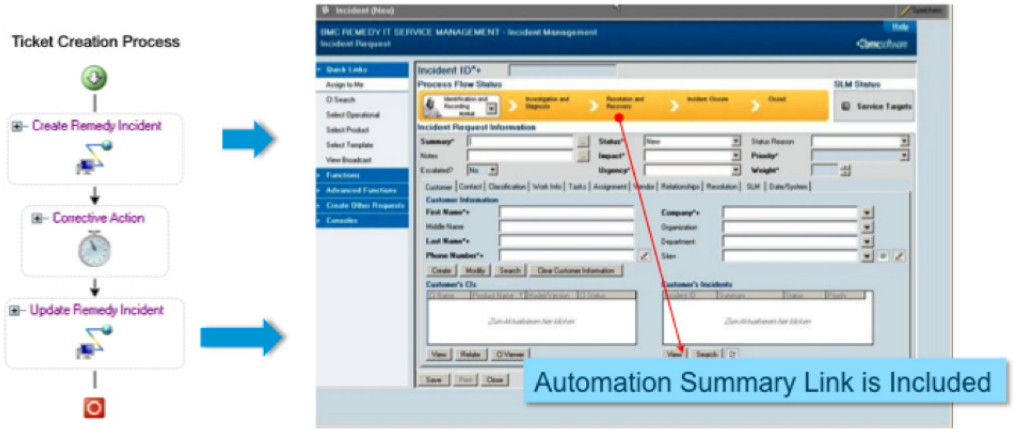
Tasks allow a process to create an IT record or perform human interaction, and provide ownership and lifecycle. Process Orchestrator provides a list of tasks in the operator console. Managers can then set or change assignments to those on their staff. Assignment changes, state changes, and all other changes are audited, so one can definitively tell exactly who did what in the system.

There are two types of tasks: IT process records and human interaction.

IT Process Records

IT process records tasks include the alerts, incidents, and change requests. These records represent the core records within IT; they are the core ITIL (IT Infrastructure Library) records. While Process Orchestrator does provide a list of these records so that one can, for example, look at a list of alerts which automation created and manage their lifecycle, more often these records are used to synchronize the records out of Process Orchestrator to a separate tool.

For example, alerts typically need to be pushed to an enterprise event manager so that the alerts can be combined with others to present a view of IT system health. Incidents and change requests are typically the domain of service desks. Rather than managing these records in Process Orchestrator, many customers prefer to manage these records elsewhere.



This construct allows processes to be independent of the event manager or service desk. For example:

1. A process such as one in a network best practice automation pack simply creates an alert, incident, or change request.
2. Optionally, this process can block waiting on the task to enter a completed state.
3. Separately, a Remedy automation pack includes a process that triggers when a new incident task is created, and creates the corresponding entry in the Remedy system.

4. The Remedy automation pack can synchronize data between the systems and when the incident reaches a terminating state in the Remedy system, the incident in Process Orchestrator is then automatically synchronized.
5. When the incident in Process Orchestrator is closed, if the process in the network best practice automation pack blocked, it can continue. For example, the process might go on to prove that the problem that generated the incident in fact no longer manifests, closing the loop to prove that the remediation did in fact fix the problem.

With this abstraction, the network best practice automation pack does not need to be dependent on the specifics of which service desk is involved, and no dependency exists between the automation packs.

Human Interactions

Human interaction tasks are steps in a workflow where a human needs to take action. They include the following types of tasks:

- Approval tasks consist of steps where the workflow needs someone's indication of what route to take. They may be simple yes or no questions, or they might give any arbitrary set of choices.
- Guided Operation tasks consist of steps in a workflow that someone must perform manually. For example, a step in a network automation workflow might be to replace a cable. While this step cannot be automated, it might be a small part of an overall workflow that can be automated.
- Review tasks allow the assignment of a document or automation summary review. When someone indicates the review is complete, the workflow can continue. Moreover, the person who indicated they completed the review is recorded in auditing. Many systems might send a notification to let someone know that they need to, for example, look at a report, but do not close the loop to see that the review actually occurred. Process Orchestrator allows closing the loop.
- Input Request tasks allow gathering data as the workflow executes.

Task Rules

Task rules provide a list of rules that act on new tasks or task changes to handle cases such as assignments, changing alert or incident severities, or setting customer-specific categories on tasks. By defining a task rule, you can tune the content you receive to your specific company without having to edit processes in the automation pack. By performing these actions by rule, one change can affect tasks created by multiple processes. For example, you can assign all SAP Incidents to a single person with a single rule.

You can view default assignments, notifications, and properties to be updated according to the following defined rule types:

Task Rules	Example
Assign a task	A new user installs and runs the SAP Automation Packs. Automation Pack processes create alerts and incidents. The user needs to assign the alerts and incidents to individuals who own that specific area of SAP. It is difficult to anticipate every way the customer may have broken up their SAP management responsibilities within the team. Virtually every customer who uses Process Orchestrator for SAP customizes the product for assignments. The user can go to a list of Task Rules and create a rule to make the assignment. In this list, they can see what other assignments have been made.
Send a task notification	Beyond notifications, when an owner is assigned, often the same or separate people must be notified in response to incidents. For example, if a job fails that is a financial job, perhaps both the technical owner of SAP job management and someone in the financial team should be notified; ownership of the incident should not be assigned just to the financial team. Note that Cisco's implementation task rules set a "people to notify" property on the task. The Task Rule identifies who to notify and a process would send the actual email.
Change alert or incident severity	Cisco's SAP automation pack assigns the severity of each detected condition when it creates the alerts and incidents. However, customer environments and requirements are unpredictable. It is very common for customers to change the severity of their alerts and incidents to match the business importance of that issue relative to others in their systems.
Add a custom category to a task	If you need to see a certain type of review, alert, or incident within a restricted view, categories are a natural way to group these alerts. Default alert categories are established in the automation pack, but users might want to add their customized categories to Tasks generated by automation packs.

Whenever a task is created, Process Orchestrator goes through the settings and conditions of each task rule that is listed and enabled. If the conditions and settings in the rule are satisfied, the task rule is executed.

Note Task rules are executed according to the order in which they are displayed in the list.

When Task Rules Execute

Task rules can not be manually run by the user. Task rules can only execute when a task is created within Process Orchestrator processes and rule conditions are met. If the task-created trigger (for any task type) and rule conditions are met, the action of the task rule is executed in the order in which it is listed in the Definitions > Task Rules view.

Any processes that are triggered based on a Task Created trigger will only execute after the appropriate task rules have executed.

Notifications

In addition to notifications that occur when a user is assigned, often the same or other people must be notified in response to a task. The task rule notification identifies which person or group to notify and a process emails the notice.

Each task contains a list of notification recipients in addition to task assignees. The notification task rule adds to this list of notification recipients. A process can react to a task create event or task change event and then appropriately notify the notification recipients by email or any other mechanism.

Automation Pack Rule Management

Task rules can be included in automation packs to easily transfer default task rules from one system to another, such as transferring task rules from a development system into a test or production system.

Some Cisco automation packs include not only processes that generate tasks, but also separate processes that handle notifications. You can create task rules to manage these objects without editing the process or task so that your customization to the tasks occurs before these notification processes run.

Triggers

Process instances can come into existence in the following ways:

- A process can be invoked manually.
- A process can be invoked by another process.
- A trigger can fire, which initiates the process.
- A process can be invoked using the northbound web service.

Triggers are events and conditions in the system that determine how or when the process will be executed. Multiple triggers can be added that can be initiated when certain conditions are met.

Process Orchestrator supports two types of rule-based triggers: events and schedules.

Events

The Process Orchestrator can monitor for events from the environment, and you can specify triggers that initiate processes when the subscribed event occurs. For example, an event might be an incoming stop trap or a fault on a UCS system.

Schedules

Schedules allow triggering processes at some time by leveraging another object called a calendar. Calendars define which days something can occur. Calendars can be selected days or sequences of dates such as weekly or monthly, they can represent dates like fiscal quarter end, or they can be combined hierarchically. Schedules then associate a time with a calendar. When the day is in the calendar, the time is evaluated. Times can be explicit or repeating (for example, hourly).

Variables

The variables feature provides a storage area for information that is used on a regular basis to avoid having to specify the same information in several places. Data stored in a variable can be altered to affect process execution behavior.

Activity Configuration

One of the most common uses of variables is to define activity configuration. Any field in an activity can refer to a variable value rather than an explicit value. For example, you can use a variable to:

- Specify the target as the machine where an event occurred
- Specify the start date of the process' operations window as a parameter to an operating system command
- Specify a condition, such as a file that should not exist after a job is initially triggered by the arrival of that file where a prior activity should have deleted the file

Process Control Components

Processes can use variables to define the control components. For example, you can use a variable to define:

- A Conditional activity to look at the exit code of a prior activity
- A While Loop activity to loop until a query fails or loop for a number of times corresponding to a number of objects pulled from a query

Process Parameters

Variables can be parameterized so that a process definition can be vague enough to be reused in multiple places and the specifics can remain undefined so that they can be defined by the person or process that invokes the process or activity.

For example, a process called notify server owner might use a variable server for the name of the server that has a problem. The process retrieves the email address of the owner of the server and sends an email.

This process might be called from multiple places; a step in a server maintenance job fails, so the maintenance job populates the Server variable and invokes the notify server owner process. Another process might notify the server owner when a backup completes.

Formulas as Variable Values

You can specify a formula anywhere a variable value is used. For example, an operating system command's parameter might be formed from concatenating two variables' values or from parsing the output of a prior command.

Adapters No Longer Supported as of Cisco Process Orchestrator 3.5

The following adapters are no longer supported and therefore instructions on how to implement them has been removed from the Online help and the user guide.

- SAP ABAP Adapter
- SAP Java Adapter
- SAP Solution Manager Adapter

- IBM DB2 Database Adapter
- JMX Adapter
- OLAP Database Adapter