

1. Introduction to Service Impact	2
1.1 Understanding service states	4
1.2 State propagation policies	6
1.3 Actual and derived state	8
2. Tutorial: Creating a service model	10
2.1 Define the service to model	11
2.1.1 Tutorial devices and member types	12
2.1.2 Introduction to the tutorial environment	14
2.2 Create logical node and subservice members for Internet connections	15
2.2.1 Create logical node	16
2.2.2 Create a subservice member for one Internet connection	17
2.2.3 Create a subservice member for the other Internet connection	18
2.2.4 Create service member to represent redundant paths to the Internet	19
2.2.5 Add a policy to the Internet connection member	20
2.3 Create members for major network segments	21
2.4 Create summary members for critical paths	22
2.5 Create a member for the network service	23
2.6 Create a member for the service model	24
2.7 Send events to fake devices	25
2.8 Remove tutorial service model members	26
3. Administering Service Impact	27
3.1 Installing or updating Service Impact	28
3.1.1 Installing Service Impact	29
3.1.1.1 Preparing to install or update Service Impact	31
3.1.2 Preparing to update to 5.5.x	32
3.1.3 Exporting all dynamic services	33
3.1.3.1 Downloading and staging export fix packages	34
3.1.3.2 Installing the export fix package	35
3.1.4 Downloading and staging 5.5.x packages	36
3.1.5 Installing Service Impact 5.5.x packages	37
3.1.6 Integrating Service Impact and Resource Manager	39
3.1.7 Updating both Service Impact ZenPacks	40
3.1.8 Rebuilding the graph database	42
3.1.8.1 Graph update tips	44
3.2 Exporting service models from one system to another	45
3.2.1 Exporting a service model	47
3.2.2 Importing service model definitions	48
3.2.2.1 Preparing to import	49
3.2.2.2 Initiating the import	50
3.2.2.3 Reconciling originating and target system entities	51
3.2.2.4 Attempting to reconcile	52
3.2.2.5 Canceling the service model reconcile	53
3.2.2.6 Adding the service models to the target system	54
3.2.2.7 Future export scenarios and machine learning	55
3.2.2.8 zenimpactimport	56
3.2.2.9 Example export file	57
3.2.2.10 Example import results file	59
3.3 Adding an Impact Services portlet	61
3.4 Production state propagation threshold	62
3.4.1 Configuring the production state propagation threshold	63
3.5 Service Impact server configuration files	64
3.6 Removing Service Impact	65
3.7 Service Impact MIB file	66
3.8 Dynamic service name restrictions	67
3.9 Configuring relationship processing	68
4. Service Impact home page	69
4.1 DYNAMIC SERVICES page	71
4.1.1 Members view	72
4.1.2 Add to Service dialog box	73
4.1.3 Impact Events	74
4.1.4 Impact View	75
4.1.4.1 Node tile	78
4.1.5 Impact Policies dialog box	79
4.1.5.1 Edit Policy options (Availability)	80
4.1.5.2 Edit Policy options (Performance)	81
4.1.5.3 Edit Custom State Provider options	82
4.2 LOGICAL NODES page	83
4.3 METATYPE CONFIGURATION page	85
5. Release notes	86
5.1 Service Impact 5.5.0 CA	87
5.2 Service Impact 5.4.8	88
5.3 Service Impact 5.4.7 CA	89
5.4 Service Impact 5.4.6	90
5.5 Service Impact 5.4.5 CA	91
5.6 Service Impact 5.3.1	93

Introduction to Service Impact

Zenoss Service Impact provides two key functions that help you reduce the time it takes to uncover and fix issues, and to eliminate unforced outages. These functions are:

Root cause analysis identifies infrastructure dependencies that are supporting a service and how status and performance issues might be affecting the overall service.

Impact analysis identifies which services are affected by a particular piece of infrastructure.

Root cause analysis

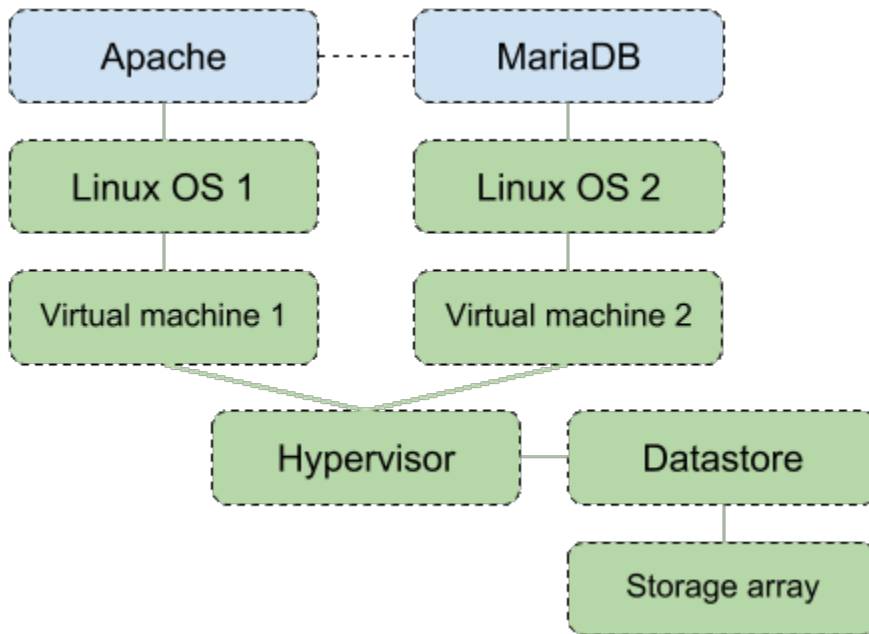
For example, consider a service we'll call Simple LAMP - a two tier web application service. The elements of the service are an Apache process serving web pages and a MariaDB database process that stores and retrieves dynamic data.

Here's a representation of Simple LAMP:



There's much more to it, of course. When we consider the infrastructure that these two applications rely on, the drawing gets more complex. The processes run on separate Linux operating systems, which might be running in virtual machines on one or more hypervisors, with backing storage provided by a storage array.

Here's a more realistic view of Simple LAMP:



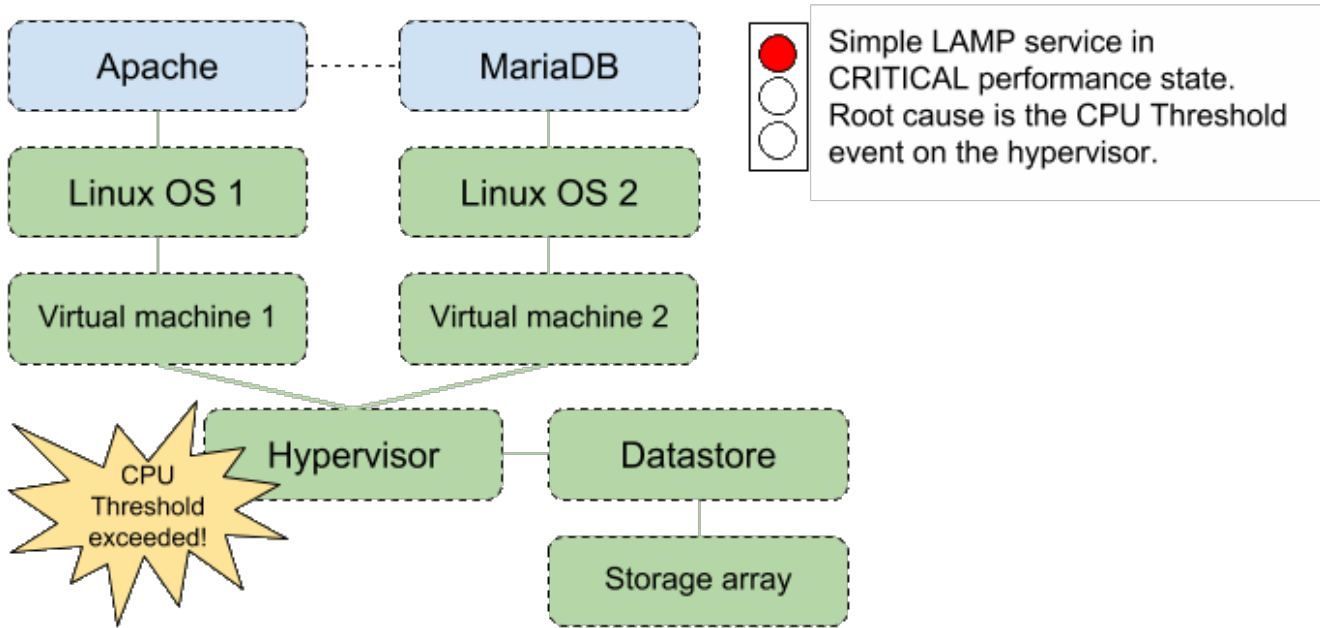
Service Impact automatically builds and maintains a model of the relationship among the service elements and their infrastructure dependencies. For the Apache and MariaDB processes, the operating systems, virtual machines, hypervisor, datastore, and storage array are all dependencies. What happens if the two virtual machines move to a new hypervisor? Or a public cloud? Service Impact automatically adjusts its model to track the proper dependencies. You don't have to make any configuration changes, as long as the new infrastructure is being monitored by Resource Manager.

If the Simple LAMP service isn't acting properly, Service Impact can help you discover where faults originate. We call this **root cause analysis**. Here's how it works.

Zenoss Resource Manager routinely collects state change data and generates threshold events where performance metrics are out of the norm. Service Impact determines which state change and threshold events apply to the Simple LAMP service—including the dependencies and generates a state for the service. Each time the state of the service changes, Service Impact generates an event that you can use to generate a service ticket, send an email, or start corrective action (see [Triggers and notifications](#)).

For example, let's pretend that the Hypervisor doesn't have enough CPU capacity to meet all the demand. Resource Manager generates a critical CPU Threshold Event for the hypervisor. Since everything in the Simple LAMP service relies on the hypervisor, Service Impact sets the state of the service to indicate that the service is in a critical performance state, and identifies the event on the hypervisor as the root cause of the event.

Here's an illustration of the hypervisor event:



Impact analysis

Impact Analysis helps teams identify whether any key services are being supported by a particular piece of infrastructure. This can prevent downtime caused by operational errors, such as forgetting to migrate virtual machines from a hypervisor before applying patches.

Once you've defined services for root cause analysis, impact analysis is performed automatically. To see which services rely on a piece of infrastructure, click on the Services view. In the following example, nine services rely on the VMware host named `ucs1-3-8-zenoss.1oc`.

Events	Health	Service
✓	A p	Database (/WebSite/OnPrem/Database)
⚠	A p	Datacenter (/Dashboards)
✓	A p	DB (/WebSite/OnPrem/Database/DB)
✓	A p	demo3 (/MHServer)
⚠	A p	OnPrem (/WebSite/OnPrem)
✓	A p	Port (/WebSite/OnPrem/Database/Port)
✓	A p	Process (/WebSite/OnPrem/Database/Process)
⚠	A p	Web Site (/Dashboards)
⚠	A p	WebSite (/WebSite)

Understanding service states

Service Impact uses Resource Manager events to determine the availability and performance states of services. From highest to lowest, the availability states are Up, Degraded, At Risk, and Down. Performance states are Acceptable, Degraded, and Unacceptable.

If there are no negative Resource Manager events for the elements or dependencies of a service, then the availability state is Up and the performance state is Acceptable. The following table summarizes the default policies for negative events:

Service state type	Event class	Event severity	Service state
Availability	/Status/* (except /Status/SNMP and /Status/WMI)	Critical	Down
		Error	Degraded
		Warning	At Risk
Performance	/Perf/*	Critical	Unacceptable
		Error	Degraded
		Warning	Degraded

Note that if an event has been generated for a component of a device (for example, a specific process), by default this event is not interpreted by Service Impact for the purposes of the device's availability or performance. In order for events related to a component to be interpreted by Service Impact, the component itself must be added to the service as an element. Administrators can identify the component that must be added by examining the component field of the event.

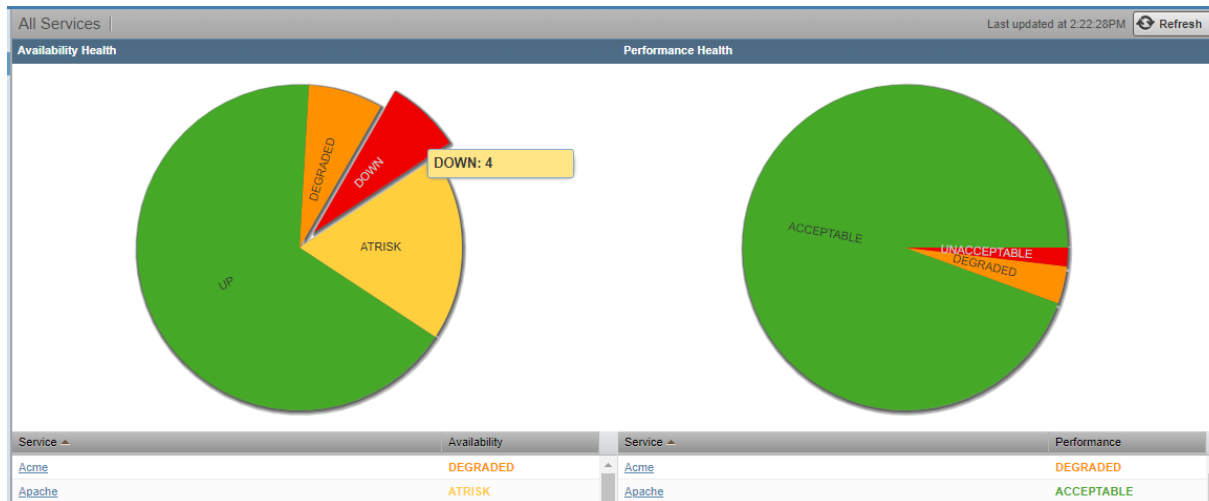
For example, assume that LinuxServerA is an element of a service called Wordpress Blog. If the HTTP process on LinuxServerA quits, a Resource Manager event is created, but Service Impact will not interpret that event for the purposes of LinuxServerA's availability or performance states, and therefore the event will not affect the availability or performance status of the Wordpress Blog service. If the administrators of Wordpress Blog decide that the HTTP process on LinuxServerA is indeed a dependency of the Wordpress Blog service, they must add the HTTP process on LinuxServerA as an element of the Wordpress Blog service in Service Impact.

Viewing service states

You can see the current service state in multiple places. Availability and performance states are displayed in a Service Health indicator. This example service is At Risk for availability and Acceptable for performance.



Service Dashboards provide a convenient way to see the status of multiple services concurrently. To see these dashboards, select Dynamic Services or any Service Organizer under the Impact tab.



Service Impact events

When a service's availability or performance state changes, Service Impact sends an event to Resource Manager. The following table summarizes the default policies for negative state changes:

Service state type	Event class	Service state	Event severity
Availability	/Service/State/Availability	Down	Critical
		Degraded	Error
		At Risk	Warning
Performance	/Service/State/Performance	Unacceptable	Critical
		Degraded	Error

You can use the [triggers and notifications](#) feature of Resource Manager to let someone know that a service has changed state, to create a service desk ticket, or to generate an automated response.

State propagation policies

State propagation policies for a dynamic service context determine how the Availability and Performance state of an entity's node and the nodes it impacts change because of events being received to that entity.

When new events occur against a service model member (device, component, component group, logical node, organizing group, or dynamic service) Service Impact identifies all of the service contexts in which that member participates. For each service context, Service Impact performs the following processing:

- Applies the combination of global, service, and node context policies to decide the state of that member in this service context.
- If the combination of policies indicate that the event should be propagated further within the service context, looks up all impact dependency relationships that exist for other nodes in the same service context. Different relationships can exist in other service contexts.
- Applies the service context policies to each of those nodes to compute their new state in the service context and decide whether to continue propagating further. Propagation of state change and event continues until all state and event propagations are completed.
- If any path through the impact graph affects the dynamic service itself, generates a service event. The service event includes details about the path from the event to the service model member through the impact graph to reach the dynamic service. Multiple paths through the graph from a single member's event might reach the dynamic service node, and multiple concurrent events to different members. Therefore, details of a service event include the union list of all service event paths and a probability ranking of which path and originating member event is the root cause.

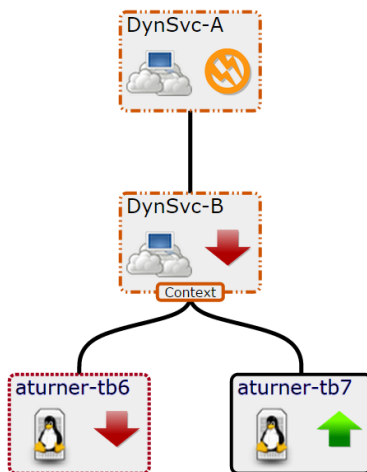
For a service model member that you use only to group child members, you can suppress sending service events.

You can create and assign multiple policies to any service member. In the Impact View, to select the type of policy that you want to create, edit, or view, select the Availability aspect or the Performance aspect.

The policy type that is applied to the members determines which members receive the data. The policy types, in order of precedence are as follows. For more information about state symbols and borders that are shown in the examples, see [Actual and derived state](#).

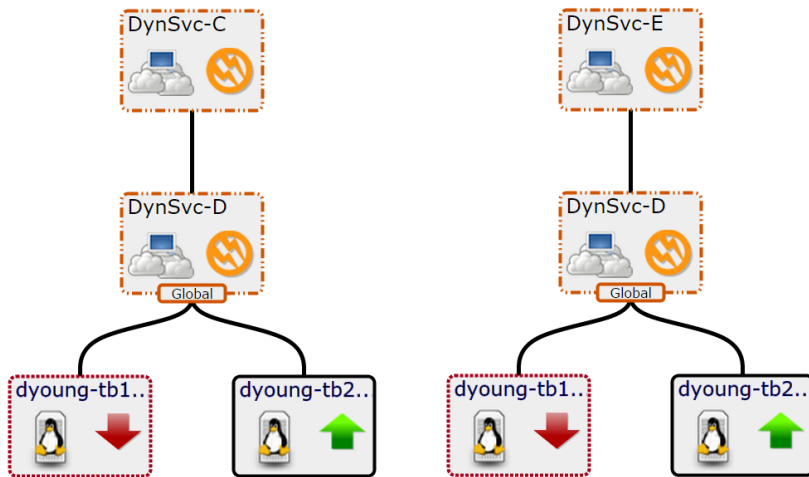
1. A contextual policy propagates the member's state change only to its immediate parent members within the current service context.

In the following example graph, a contextual policy is applied to the service model for Dynamic Service B (DynSvc-B). The policy states that if one child member is down, then the state of DynSvc-B must be degraded.



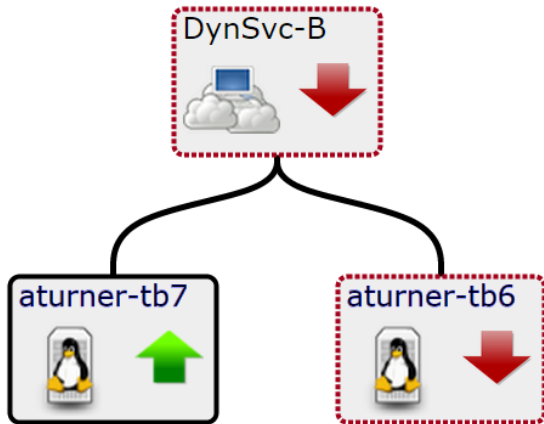
2. A global policy applies to all service model contexts that share a member that has a changed state. For example, if a global policy is applied to a member in a service model, and a child member has a change to its state data, the new state is propagated to the parent members in all service models to which the member belongs.

The following example graph shows that DynSvc-D is used in two different service models: DynSvc-C and DynSvc-E. The global policy propagates the degraded state to all service contexts of which DynSvc-D is a member.



3. A member with the default policy sends the state data of the worst condition affecting it to its parent member. The default policy is negated if you add either a contextual or global policy to a service model.

In the following example graph, no policy is applied to DynSvc-B. Its state is down because that is the worst case of its children.



If multiple policies are applied to a member, then the following overrides occur:

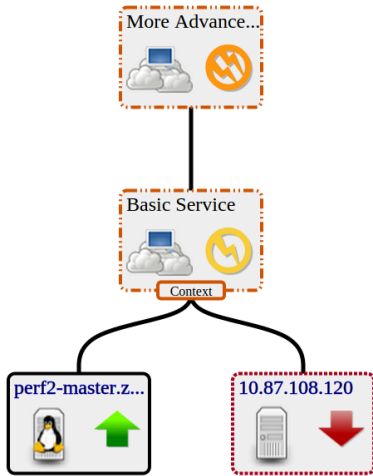
- A contextual policy overrides a global policy.
- Contextual and global policies override the default policy.

Actual and derived state

Each in a service model has an actual state and a derived state.






A device element's actual state is determined by events that occur on the device, regardless of the service models in which it participates. A service model member's actual state is generated from the service model in its own service context. The element's derived state is generated from policy propagation within a given context. In the Service Impact graph, the symbol that appears inside the node's border reflects the actual state; the border that outlines the node reflects the derived state.



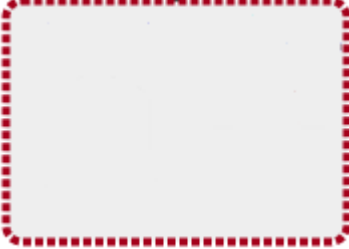
In the following example graph, the actual and derived states are down. The parents of the device have a degraded and at-risk actual state, yet the derived state based on the contextual policy, is degraded.



Visual state indicators

The Impact View provides symbols and borders to visually indicate availability, actual state, performance, and derived state, as shown in the following table.

Availability	Performance	Symbol / Actual State	Border / Derived State
UP	ACCEPTABLE		
ATRISK	Not applicable.		
DEGRADED	DEGRADED		

			
DOWN	UNACCEPTABLE		

A member's actual and derived states can be different. As shown in the following example, a service's actual state can be up, but an applied context policy changed its derived state to down.



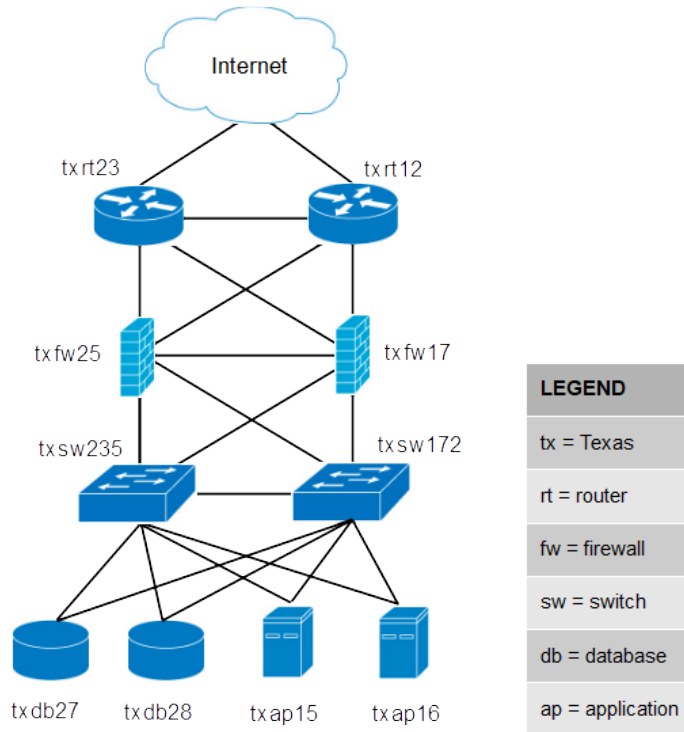
Tutorial: Creating a service model

This tutorial demonstrates how to create a service model for a simplified CRM application, view the model, and view and interpret the root-cause analyses that Service Impact creates when events affect the application's availability state.

About this tutorial:

- For information about Service Impact browser interface elements, see [Service Impact home page](#).
- Complete the tasks in the order in which they are presented.
- Where indicated, steps depend on the version of Resource Manager with which Service Impact is deployed.
- Device and component resources for this tutorial are fake, and do not affect production environments. The final task erases the fake devices and the nodes that you create during the tutorial. For more information, see [Tutorial devices and member types](#).

The following diagram shows the network topology of the CRM application that is featured in this tutorial.



Define the service to model

The service to model is a CRM application, and it is defined by the members (resources) on which it relies.

The CRM application relies on application and database hosts and processes, and on the network infrastructure that connects the service with its users. This tutorial creates a service model of the development deployment of the CRM application, not the production or quality assurance deployments.

The following procedure loads the fake devices into Resource Manager and sets up the tutorial environment.

1. Gain access to the Resource Manager CLI environment in a Zope container.
 - a. Log in to the Control Center master host as a user with serviced CLI privileges.
 - b. Start an interactive session in a Zope container as user zenoss.

```
serviced service attach zope/0 su - zenoss
```

2. Create a soft link to the Service Impact scripts directory.
 - a. Create a variable for the path of the ZenPacks.zenoss.Impact ZenPack.

```
my_zp=$(zenpack --list | awk '/\.Impact-/ { print substr($2,2,length($2)-2) }') && echo $my_zp
```

- b. Create the link.

```
ln -s $my_zp/ZenPacks/zenoss/Impact/scripts/tutorial ${HOME}/impact_scripts
```

3. Change directory to the Service Impact scripts directory.

```
cd ${HOME}/impact_scripts
```

4. Load fake devices and components into Resource Manager.

```
zenbatchload --nomodel ./devices.txt
```

5. In tutorial.sh, ensure that the values of the ZENOSS_USERNAME and ZENOSS_PASSWORD variables specify the user name and password of a Resource Manager user account, and if not, change them.
6. Set up the tutorial environment.

```
./tutorial.sh
```

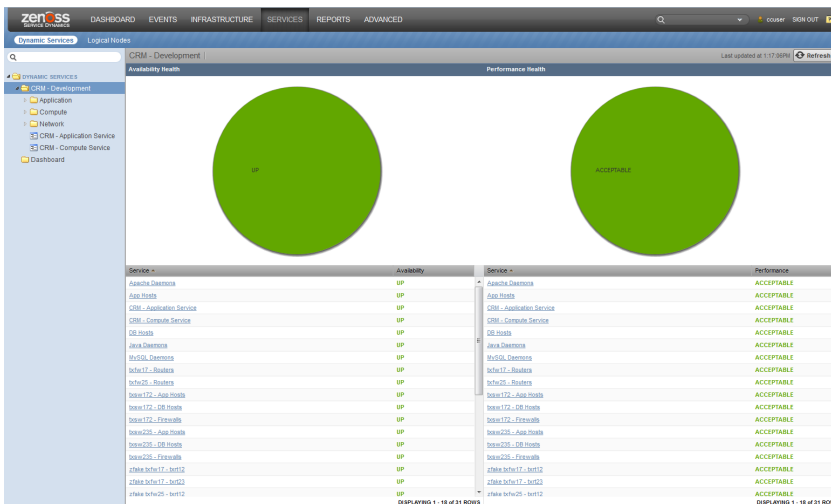
7. Log in to the Resource Manager browser interface as a user with ZenManager or Manager privileges.

8. Click SERVICES.

The Service Impact feature of Resource Manager adds a tab named SERVICES to the Resource Manager menu bar.

9. In the tree view, open the CRM - Development organizer, and then open the Application and Compute organizers.

The tree view displays the service members created by the tutorial.sh script, as shown in the following example.



Tutorial devices and member types

The tutorial uses the following fake devices and service model member types (Resource Manager device or component).

Application host 15

Device name	Description	Member type
fake-txap15	Application host 15	Device
fake-txap15-httpd	Apache daemon	Component
fake-txap15-java	Java/JRE daemon	Component
fake-txap15-nic-0	Network interface card 0	Component
fake-txap15-nic-1	Network interface card 1	Component

Application host 16

Device name	Description	Member type
fake-txap16	Application host 16	Device
fake-txap16-httpd	Apache daemon	Component
fake-txap16-java	Java/JRE daemon	Component
fake-txap16-nic-0	Network interface card 0	Component
fake-txap16-nic-1	Network interface card 1	Component

Database host 27

Device name	Description	Member type
fake-txdb27	Database host 27	Device
fake-txdb27-mysqld	MySQL daemon	Component
fake-txdb27-nic-0	Network interface card 0	Component
fake-txdb27-nic-1	Network interface card 1	Component

Database host 28

Device name	Description	Member type
fake-txdb28	Database host 28	Device
fake-txdb28-mysqld	MySQL daemon	Component
fake-txdb28-nic-0	Network interface card 0	Component
fake-txdb28-nic-1	Network interface card 1	Component

Firewall 17

Device name	Description	Member type
fake-txfw17	Firewall 17	Device
fake-txfw17-10g-0	Port 0 (10GB capacity)	Component
fake-txfw17-10g-1	Port 1 (10GB capacity)	Component
fake-txfw17-10g-2	Port 2 (10GB capacity)	Component
fake-txfw17-10g-3	Port 3 (10GB capacity)	Component
fake-txfw17-1g-0	Port 0 (1 GB capacity)	Component

Firewall 25

Device name	Description	Member type
fake-txfw25	Firewall 25	Device
fake-txfw25-10g-0	Port 0 (10GB capacity)	Component

fake-txfw25-10g-1	Port 1 (10GB capacity)	Component
fake-txfw25-10g-2	Port 2 (10GB capacity)	Component
fake-txfw25-10g-3	Port 3 (10GB capacity)	Component
fake-txfw25-1g-0	Port 0 (1 GB capacity)	Component

Router 12

Device name	Description	Member type
fake-txrt12	Router 12	Device
fake-txrt12-100g-0	Port 0 (100GB capacity)	Component
fake-txrt12-10g-0	Port 0 (10GB capacity)	Component
fake-txrt12-10g-1	Port 1 (10GB capacity)	Component
fake-txrt12-1g-0	Port 0 (1GB capacity)	Component

Router 23

Device name	Description	Member type
fake-txrt23	Router 23	Device
fake-txrt23-100g-0	Port 0 (100GB capacity)	Component
fake-txrt23-10g-0	Port 0 (10GB capacity)	Component
fake-txrt23-10g-1	Port 1 (10GB capacity)	Component
fake-txrt23-1g-0	Port 0 (1GB capacity)	Component

Switch 172

Device name	Description	Member type
fake-txsw172	Switch 172	Device
fake-txsw172-10g-0	Port 0 (10GB capacity)	Component
fake-txsw172-10g-1	Port 1 (10GB capacity)	Component
fake-txsw172-1g-0	Port 0 (1GB capacity)	Component
fake-txsw172-1g-1	Port 1 (1GB capacity)	Component
fake-txsw172-1g-2	Port 2 (1GB capacity)	Component
fake-txsw172-1g-3	Port 3 (1GB capacity)	Component
fake-txsw172-1g-4	Port 4 (1GB capacity)	Component

Switch 235

Device name	Description	Member type
fake-txsw235	Switch 235	Device
fake-txsw235-10g-0	Port 0 (10GB capacity)	Component
fake-txsw235-10g-1	Port 1 (10GB capacity)	Component
fake-txsw235-1g-0	Port 0 (1GB capacity)	Component
fake-txsw235-1g-1	Port 1 (1GB capacity)	Component
fake-txsw235-1g-2	Port 2 (1GB capacity)	Component
fake-txsw235-1g-3	Port 3 (1GB capacity)	Component
fake-txsw235-1g-4	Port 4 (1GB capacity)	Component

Introduction to the tutorial environment

The tutorial.sh script creates the following service model members and organizers to initialize the CRM service model in Service Impact. Use of these members and organizers is a recommended best practice.

- Dashboard organizer; root-level organizer for service models as a whole. Initially, the organizer is empty.
- Root-level CRM - Development organizer that contains additional organizers. Use a single root-level organizer to contain the subservices of each service model, and use standardized names (and contents) for sub-organizers.
- CRM - Application Service and CRM - Compute Service service model members, children of the CRM - Development organizer.

These members summarize the application and compute services that are associated with the CRM application. They are easily located without having to open the organizers in which their constituent subservices are located.

- Members in the Network organizer that start with zfake represent the network connections between fake devices.

Because these fake devices are not modeled, Resource Manager cannot discern their relationships, and Service Impact cannot create device or component members. Therefore, the script creates members to represent the connections.

These members have neither contextual nor global policies, so the default policy applies. That is, the state of the worst condition that affects child members becomes the state of the zfake members, which is the correct policy for these connections.

- DNS and interface names that follow a naming convention.
- Subservice members in the Network organizer that start with tx, which contain redundant resources. Standardized, global availability policies are defined.

These subservice members demonstrate the following **best practices**:

- Each subservice member contains homogeneous child members. Global policies work best when child members are homogeneous.
- Each subservice uses global policies. Global policies can be re-used across service model boundaries. Contextual policies are restricted to specific service models.
- Each global policy contains the following, standardized availability state triggers: ATRISK if 50% or more child members are down; DOWN if 100% of child members are down. Use of percentage thresholds means that the policies do not need to be adjusted if additional resources are deployed later. Note: The standardized state triggers that are used in this case are examples of systematically thinking about and using global policies, and not intrinsically best practices. For example, if a resource pool contains more than two members, additional state triggers can be defined.

The remaining procedures in this tutorial demonstrate how to complete and test the CRM service model.

Create logical node and subservice members for Internet connections

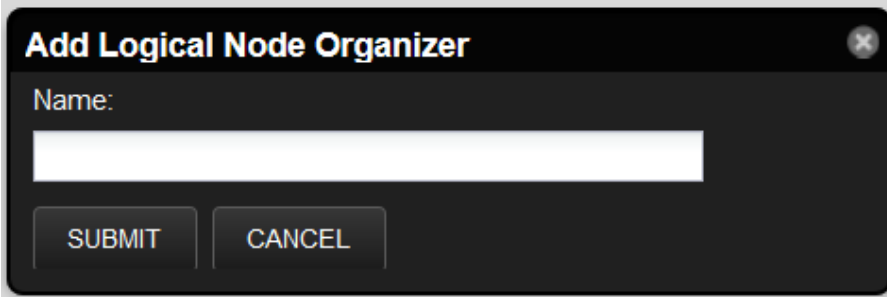
All network connections except the connection to the Internet are modeled. Because the Internet connection is not modeled in Resource Manager, we create a logical node to represent it. For this tutorial, the logical node is simply a service model member that reacts to fake events. A "real" logical node could be configured to react to events from a zencommand that pings an Internet resource.

- [Create logical node](#)
- [Create a subservice member for one Internet connection](#)
- [Create a subservice member for the other Internet connection](#)
- [Create service member to represent redundant paths to the Internet](#)
- [Add a policy to the Internet connection member](#)

Create logical node

Follow these steps:

1. In the Resource Manager browser interface, select SERVICES > Logical Nodes.
2. From the Add menu at the bottom of the tree view, select Add Logical Node Organizer.



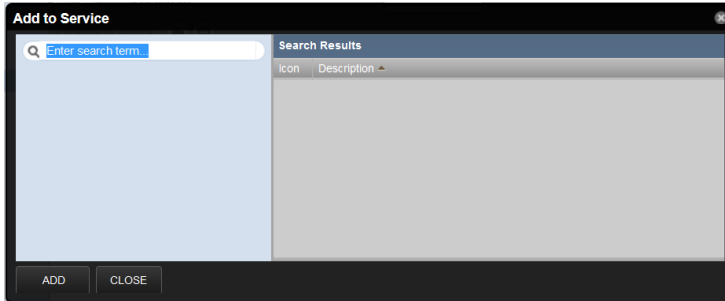
3. In the Add Logical Node Organizer dialog box, enter CRM - Development, and then click SUBMIT.
4. From the Add menu at the bottom of the tree view, select Add Logical Node.
5. In the Add Logical Node dialog box, enter example.com, and then click SUBMIT.
6. In the example.com details view, enter values to match the following table, and then click Save.

Field	Description
Description	A node to help represent a route to the Internet.
Criteria	all, Summary, contains, fakeInternet
Events for this node in this event class	/Status/Ping
will result in these availability states	Critical: DOWN, Error: DOWN, Warning: ATRISK, Clear: UP

Create a subservice member for one Internet connection

Create a service model member for the connection from router 12 to the zenoss.com logical node.

1. In the Resource Manager browser interface, select SERVICES > Dynamic Services.
2. In the tree view of organizers, open CRM - Development > Network.
Unless an organizer is selected, new members are created at the root level. From there, you can drag the new member into the correct organizer.
3. From the Add menu at the bottom of the tree view, select Add Dynamic Service.
4. In the Add Dynamic Service dialog box, enter txrt12 - Internet, and then click SUBMIT.
5. In the Members view, click Add.
6. In the Add to Service search field, enter fake-txrt12.



7. In the left column, select Device, and in the results list, select fake-txrt12-100g-0, and then click ADD.
8. In the search field, enter example.com.
9. From the search results list, select example.com, and then click ADD > CLOSE.

Create a subservice member for the other Internet connection

To create the service model member for the connection through router 23, clone the member for the connection through router 12.

1. In the tree view, select service model member txrt12 - Internet.
2. From the Action menu at the bottom of the tree view, select Clone Service.
3. In the Clone Service dialog box, enter txrt23 - Internet, and then click SUBMIT.
The new service is created and its contents are displayed in the Members view.
4. From the list of members in the Members view, select fake-txrt12-100g-0, and then click Remove.
If you click on the name of the member, Resource Manager displays the device overview page. To return to the correct page, click the browser's back button.
5. In the Remove Items dialog box, click OK.
6. In the Members view, click Add.
7. In the Add to Service search field, enter 100g.
8. In the left column, select Device.
9. In the results list, double-click fake-txrt23-100g-0, and then click ADD > CLOSE.

Create service member to represent redundant paths to the Internet

The previous tasks created subservices to represent the WAN connections to the Internet. This task creates a subservice member in the service model to represent the redundant WAN tier.

1. In the tree view, select Network.
2. From the Add menu, select Add Dynamic Service.
3. In the Add Dynamic Service dialog box, enter Routers - Internet, and then click SUBMIT.
4. In the Members view, click Add.
5. In the Add to Service search field, enter Internet.
6. In the left column, select DynamicService.
7. In the results list, select txt12-Internet and txt23-Internet, and then click ADD > CLOSE.

Add a policy to the Internet connection member

Add a global availability policy to the Internet connection subservice member.

1. In the tree view, select Routers - Internet.
2. From the view menu, select Impact View.
3. In the node tile of the Routers - Internet service (the tile at the top of the hierarchy), right-click and then select Edit Impact Policies.
4. In the Impact Policies dialog box, directly to the right of the Global Policy entry, click Add.



5. In the lower-left corner of the Edit...Policy tab, click Add, and then add the following triggers.
 - ATRISK if $\geq 50\%$ DynamicService is DOWN
 - DOWN if $\geq 100\%$ DynamicService is DOWN

When you select a type for a state trigger, the selection excludes other types. In this case, the only options are Any and DynamicService, and both child members are service members, so exclusivity is not a concern.

Create members for major network segments

All connections between devices and all redundant resources are modeled. This procedure creates service model members for the major network segments.

1. In the Resource Manager browser interface, select SERVICES > Dynamic Services.
2. In the tree view, open CRM - Development > Network.
3. Create new members for the major network segments. For instructions, see preceding steps.
The following table shows new and existing members.

New service model member	Existing service model members
App Hosts - Switches	txsw235 - App Hosts, txsw172 - App Hosts
DB Hosts - Switches	txsw235 - DB Hosts, txsw172 - DB Hosts
Firewalls - Routers	txfw25 - Routers, txfw17 - Routers
Switches - Firewalls	txsw235 - Firewalls, txsw172 - Firewalls

4. Add the following global policies (availability state triggers) to each new service model member. For instructions, see preceding steps.
 - ATRISK if >= 50% DynamicService is DOWN
 - DOWN if >= 100% DynamicService is DOWN

Create summary members for critical paths

Service model members for all of the major network segments are in place. Now consider members to represent the critical paths. To characterize the CRM application as available, the following minimum connections are required:

- One connection between Internet users and an application server that is UP.
- One connection between an application server and a database server that is UP.

If entities for these paths existed, we could create the service model member that summarizes the network service for CRM. However, only one segment is defined, by the Routers - Internet subservice. The following paths require an additional subservice each:

- The path between the routers and the application hosts.
- The path between the application and database hosts.

1. In the Resource Manager browser interface, select SERVICES > Dynamic Services.
2. In the tree view, open the CRM - Development organizer, and then open and select the Network organizer.
3. Create new service model members for the network paths that affect users.

Because each subservice is critical, the correct policy for these new service model members is the default policy. The following table matches new and existing service model members.

New service model member	Existing service model members
Routers - App Hosts	App Hosts - Switches, Firewalls - Routers, Switches - Firewalls
App Hosts - DB Hosts	App Hosts - Switches, DB Hosts - Switches

For detailed instructions, see the preceding steps.

Create a member for the network service

The critical network paths are defined. Create a service model member to represent the network service for the CRM application.

1. In the Resource Manager browser interface, select SERVICES > Dynamic Services.
2. In the tree view, open the CRM - Development organizer, and then select it.
Service model members that represent a category should be peers of their sub-organizers.
3. Create a new service model member, named CRM - Network Service.
4. Add the following subservice model members to the new service model member.
 - App Hosts - DB Hosts
 - Routers - App Hosts
 - Routers - Internet

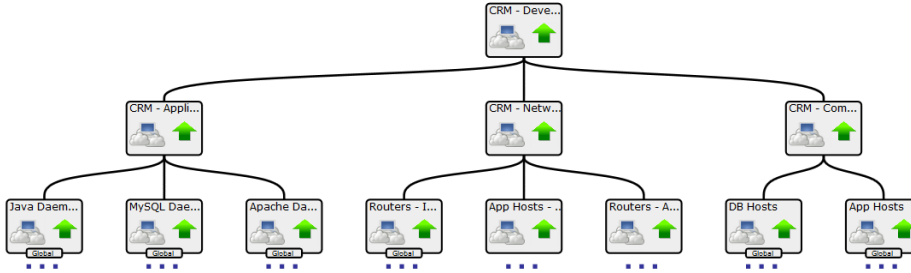
The correct policy for this member is the default policy.

Create a member for the service model

After all resource categories are modeled, the application as a whole can be modeled.

1. In the Resource Manager browser interface, select SERVICES > Dynamic Services.
2. In the tree view, select the Dashboard organizer.
Keep service model members that represent a service model as a whole in a separate, root-level organizer. This way, you can quickly determine the state of all service models in the environment.
3. Create a service model member named CRM - Development Service and add the following subservice model members to it.
 - CRM - Application Service
 - CRM - Compute Service
 - CRM - Network Service

To view the graph of the service model, select the new service in the tree view, and then select Impact View. After hiding child nodes and zooming in, the graph looks similar to the following image. If necessary, to reposition the graph, click and drag it.



Send events to fake devices

To see how they affect the availability of the CRM application, send events to the fake devices.

1. In the tree view, open the Dashboard organizer, and then select CRM - Development Service.
2. In the main view area, select Impact View.
3. Gain access to the Resource Manager CLI environment in the ZenHub container.
 - a. Log in to the Control Center master host as a user with `serviced` CLI privileges.
 - b. Start an interactive session in a Zope container as the `zenoss` user.

```
serviced service attach zenhub su - zenoss
```

4. Send ping down events to the fake network interface card components in the `txap15` and `txap16` hosts.

```
zensendevent -d fake-txap15-nic-0 -c /Status/Ping -s Critical "Impact Tutorial - fake device is DOWN"
zensendevent -d fake-txap15-nic-1 -c /Status/Ping -s Critical "Impact Tutorial - fake device is DOWN"
zensendevent -d fake-txap16-nic-0 -c /Status/Ping -s Critical "Impact Tutorial - fake device is DOWN"
```

5. In the browser, click Refresh.

The Impact View shows that the CRM application availability state changed to ATRISK, and the nodes involved in the state change are highlighted with yellow and red.

To see all nodes in the Service Impact graph, click Expand All.

6. Change the view to Impact Events, and then in the Impact Events list, click CRM - Development Service.

Status	Severity	Service	Event Class	Summary	First Seen	Last Seen	Count
Warning	Critical	CRM - Development Service	/Service/State/Availability	Service Dashboard/CRM - Development Service is ATRISK.	2013-10-24 12:37:05	2013-10-24 12:37:05	1

Confidence	Status	Severity	Resource	Component	Event Class	Summary	First Seen	Last Seen	Count
38	Down	Critical	fake-txap16-nic-0	/Status/Ping	Impact Tutorial - fake device is DOWN	2013-10-24 12:37:05	2013-10-24 12:37:05	1	
38	Down	Critical	fake-txap15-nic-1	/Status/Ping	Impact Tutorial - fake device is DOWN	2013-10-24 12:37:04	2013-10-24 12:37:04	1	
26	Down	Critical	fake-txap15-nic-0	/Status/Ping	Impact Tutorial - fake device is DOWN	2013-10-24 12:37:04	2013-10-24 12:37:04	1	

The events that contribute to the current state of the CRM application are weighted by the root-cause analysis that Service Impact performs (the Confidence column). To view the impact chain of an event, click the plus button in the left column.

7. Send ping down events to the fake network interface card components in the `txdb27` and `txdb28` hosts, and then click Refresh.

```
zensendevent -d fake-txdb27-nic-0 -c /Status/Ping -s Critical "Impact Tutorial - fake device is DOWN"
zensendevent -d fake-txdb27-nic-1 -c /Status/Ping -s Critical "Impact Tutorial - fake device is DOWN"
zensendevent -d fake-txdb28-nic-0 -c /Status/Ping -s Critical "Impact Tutorial - fake device is DOWN"
```

The list of contributing events grows, and the rankings change to reflect the added events.

8. Send clear events to all fake devices that are down.

```
zensendevent -d fake-txap15-nic-0 -c /Status/Ping -s Clear "Impact Tutorial - fake device is UP"
zensendevent -d fake-txap15-nic-1 -c /Status/Ping -s Clear "Impact Tutorial - fake device is UP"
zensendevent -d fake-txap16-nic-0 -c /Status/Ping -s Clear "Impact Tutorial - fake device is UP"
zensendevent -d fake-txdb27-nic-0 -c /Status/Ping -s Clear "Impact Tutorial - fake device is UP"
zensendevent -d fake-txdb27-nic-1 -c /Status/Ping -s Clear "Impact Tutorial - fake device is UP"
zensendevent -d fake-txdb28-nic-0 -c /Status/Ping -s Clear "Impact Tutorial - fake device is UP"
```

Remove tutorial service model members

This step removes all tutorial-defined members.

1. Log in to the Resource Manager browser interface as a user with ZenManager or Manager privileges.
2. Click SERVICES.
3. Remove the dynamic services and logical node.
 - a. In the tree view, select the Dashboard organizer.
 - b. At the bottom of the tree view, click Delete.
 - c. In the tree view, select CRM - Development.
 - d. At the bottom of the tree view, click Delete.
 - e. Select the Logical Nodes view mode.
 - f. In the tree view, select CRM - Development.
 - g. At the bottom of the tree view, click Delete.
4. Remove the fake devices.
 - a. In the Resource Manager browser interface, select INFRASTRUCTURE.
 - b. In the tree view, select FakeImpactDevices.
 - c. At the bottom of the tree view, click Delete.

Administering Service Impact

This section includes administration procedures.

- [Installing or updating Service Impact](#)
- [Exporting service models from one system to another](#)
- [Adding an Impact Services portlet](#)
- [Production state propagation threshold](#)
- [Service Impact server configuration files](#)
- [Removing Service Impact](#)
- [Service Impact MIB file](#)
- [Dynamic service name restrictions](#)
- [Configuring relationship processing](#)
- [Hidden page holding area](#)

Installing or updating Service Impact

Service Impact is packaged as a Docker image and two ZenPacks: ZenPacks.zenoss.Impact and ZenPacks.zenoss.ImpactServer. The files are available from delivery.zenoss.com. The ZenPacks require customized installation and update procedures, documented here.

The `Impact` service includes the Service Impact server and database. The install process adds the service to the Infrastructure hierarchy of Resource Manager. Its requirements are minimal, compared to the other services in that hierarchy. The resource pool you select for the Infrastructure hierarchy easily accommodates the Impact service.

The `zenimpactstate` service includes the `zenimpactstate` daemon. The service is included in the Events category of the Zenoss hierarchy of Resource Manager.

Once Service Impact is installed, Resource Manager is dependent on it. If Service Impact services are unavailable, Resource Manager continues to monitor devices, but is unable to perform modeling, or properly install or remove ZenPacks.

For optimum results, review installation or update procedures before starting either process.

Keep this page open, and open new tabs or windows for each update procedure.

Installing Service Impact

1. [Prepare to install](#)
2. [Install Service Impact](#)

Updating 5.1.x, 5.2.x, or 5.3.x to 5.5.0

Use the following procedures to update Service Impact from a previous minor release to release 5.5.0:

1. [Prepare to update](#)
2. [Export all dynamic services](#)
3. [Download and stage packages](#)
4. [Remove the previous version](#)
5. [Install Service Impact packages](#)
6. [Integrate Service Impact and Resource Manager](#)

Updating 5.4.x to 5.5.0

Use the following procedures to update Service Impact from 5.4.x to 5.5.0.

Before performing an update:

- [Verify that Resource Manager is running normally](#). The update procedures stop, start, or restart specific services, but the required starting point is Resource Manager running normally.
- [Export service models](#). Exporting before an update provides a quick restore option if the update fails. If necessary, you can uninstall, remove the ZenPack, install the working version, and then import the service models.

To update:

1. Update [DynamicView](#) to v1.7.1 or a more recent version:
 - a. [Prepare to update the ZenPack](#)
 - b. [Update the ZenPack](#)
2. Update to Service Impact v5.5.0:
 - a. [Prepare to update](#)
 - b. [Update both ZenPacks](#)
3. [Rebuild the graph database](#)

Installing Service Impact

Before performing this procedure, complete the steps in [Preparing to install or update Service Impact](#).

1. Log in to the Control Center master host as a user with Control Center CLI privileges.
2. Create a snapshot:

```
serviced service snapshot Zenoss.resmgr
```

On completion, the `serviced` command returns the ID of the new snapshot. If the installation of a ZenPack fails, you can restore the snapshot created in this step. For more information about restoring a snapshot, see [Creating snapshots and rolling back](#).

3. Stop the Zenoss services, and then verify that the services are stopped.
 - a. Stop Resource Manager.

```
serviced service stop Zenoss.resmgr/Zenoss
```

- b. Wait until all services are stopped.
Use the `watch` command to monitor the status.

```
watch serviced service status Zenoss.resmgr/Zenoss
```

4. Start the `zeneventserver` service.
 - a. Start `zeneventserver`.

```
serviced service start zeneventserver
```

- b. Wait until the service is started.
Use the `watch` command to monitor the status.

```
watch serviced service start zeneventserver
```

5. Install the `ZenPacks.zenoss.ImpactServer` ZenPack.
 - a. Change directory to the directory in which the Service Impact ZenPack egg files are located.
For example, the `/tmp/impact` directory:

```
cd /tmp/impact
```

- b. Install the ZenPack.
Replace `VERSION` with the ZenPack version number:

```
serviced service run zope zenpack-manager install ZenPacks.zenoss.ImpactServer-VERSION-py2.7.egg
```

6. Start the Infrastructure/Impact service, and then verify that it started.
 - a. Start the Impact service.

```
serviced service start Infrastructure/Impact
```

- b. Verify that the service is started.

```
watch serviced service status Infrastructure/Impact
```

7. Install the `ZenPacks.zenoss.Impact` ZenPack.
Replace `VERSION` with the ZenPack version number:

```
serviced service run zope zenpack-manager install ZenPacks.zenoss.Impact-VERSION-py2.7.egg
```

8. Identify the delegate hosts where Resource Manager is running.

```
serviced host list --show-fields=Pool,Name
```

The name of the resource pool where Resource Manager is running is not standardized.

9. Log in to a delegate host, and then start a Zope container.
 - a. Log in to a delegate host in the Resource Manager pool as root or as a user with `serviced` CLI privileges.
 - b. Start a Zope container on the host.

```
serviced service shell zope
```

- c. In the container, log in as user zenoss.

```
su - zenoss
```

10. Update the catalog and then initialize the server.

- a. Update the catalog.

```
zenimpactgraph run -x catclean
```

- b. Create the Service Impact graph in ZODB and Neo4j.

```
zenimpactgraph run --update
```

Processing may appear stuck for tens of minutes as connected objects are found in the ZODB. This is normal.

- c. Exit the zenoss user account.

```
exit
```

- d. Exit the Zope container.

```
exit
```

11. Restart all Zenoss services:

```
serviced service restart Zenoss.resmgr/Zenoss
```

Preparing to install or update Service Impact

This section describes how to prepare to install or update Service Impact, and provides instructions for downloading and staging required software.

The following list outlines recommended best practices for installing or upgrading Service Impact:

- Review the [release notes](#) for this release before proceeding. The latest information is provided there.
- Use [screen](#), [tmux](#), or a similar program to establish sessions on the master host. If you become disconnected, the commands you initiate will continue to run.
- Review the procedures in this section before performing them. Every effort is made to avoid mistakes and anticipate needs; nevertheless, the instructions may be incorrect or inadequate for some requirements or environments.

Downloading required files

To perform this procedure, you need:

- A workstation with internet access.
- Permission to download files from [delivery.zenoss.com](#). To request permission, file a ticket at the [Zenoss Support](#) site.
- The names of the files to download. The files are listed in the [release notes](#).
- A secure network copy program.

Follow these steps:

1. In a web browser, navigate to the download site, and then log in.
The download site is [delivery.zenoss.com](#).
2. Download the Docker image file for Service Impact.
 - `install-zenoss-impact_VERSION.run`
Replace Version with the most recent version number available on the download page.
3. Download the Service Impact ZenPack files.
 - `ZenPacks.zenoss.ImpactServer-VERSION-py2.7.egg`
 - `ZenPacks.zenoss.Impact-VERSION-py2.7.egg`
Replace Version with the most recent version number available on the download page.
4. Use a secure copy program to copy the Docker image file and the ZenPack files to the Control Center master host.

Staging required files on the master host

To perform this procedure, you need permission to log in to the Control Center master host as root, or as a user with superuser privileges. Use this procedure to install the Docker image and to prepare the ZenPack files for installation.

1. Log in to the Control Center master host as root, or as a user with superuser privileges.
2. Stage the Docker image file and ZenPack files in `/tmp`.
 - a. Create a directory in `/tmp` for the files. The directory must be local (not mounted) and must be readable, writable, and executable by all users. For example, `/tmp/impact`.

```
mkdir /tmp/impact
```

- b. Copy or move the Docker image file and ZenPack files to `/tmp/impact`.

Do not change the file names during the move or copy.

- c. Change the file permissions. The files must have the same permissions as their parent directory.

```
chmod -R 777 /tmp/impact
```

3. Install the Service Impact image.
 - a. Change to the directory in which the Service Impact image is located.

```
cd /tmp/impact
```

- b. Install the image.

```
yes | ./install-zenoss-impact_*.run
```

[Like](#)

Preparing to update to 5.5.x

This section includes steps to perform before starting the process of updating Service Impact.

1. Update Resource Manager to 6.3.x or a later release.
Previous releases of Resource Manager do not include features that Service Impact 5.5.x requires.
2. Update ZenPacks.zenoss.DynamicView to v1.7.1 or a more recent version:
 - a. [Prepare to update the ZenPack](#)
 - b. [Update the ZenPack](#)
3. Verify that Resource Manager is operating normally.
For more information, see [Preparing a deployment for update](#).

Perform the remaining procedures during a scheduled maintenance window.

Exporting all dynamic services

The 5.5.x update process requires a successful export of all existing dynamic services, if any exist. If not, you can skip this step.

Follow these steps to export the services:

1. In Resource Manager, navigate to SERVICES > Dynamic Services.
2. In the tree view of organizers, select DYNAMIC SERVICES.
3. From the tool menu at the bottom of the tree view, select Export Selected.
4. Verify that the export file downloaded to your workstation successfully.

If you get an error such as the following, the export failed:

A Zenoss error has occurred

[View Error Details](#)

An error was encountered while publishing this resource. Please use the form below to submit the error details. The Zenoss [community forums](#) are very active and a good resource for solving problems and providing feedback. The following fields are optional. This information will only be used to contact you if further information is needed.

Your name:

Your email address:

Additional info you would like to provide:

Click this button to send the above information to Zenoss, Inc.

If the export fails, perform the following procedures before continuing:

- [Downloading and staging export fix packages](#)
- [Installing the export fix package](#)

When you complete the procedures, return to the steps at the beginning of this page and export the services.

Downloading and staging export fix packages

To perform this procedure, you need:

- A workstation with internet access.
- Permission to download files from delivery.zenoss.com. To request permission, file a ticket at the [Zenoss Support](#) site.
- A secure network copy program.

Follow these steps:

1. In a web browser, navigate to the download site, and then log in.
The download site is delivery.zenoss.com.
2. Download the following files:
 - `install-zenoss-impact_5.3_5.3.4.0.1_34.run`
 - `ZenPacks.zenoss.ImpactServer-5.3.4.0.1-75-py2.7.egg`
3. Use a secure copy program to copy the files to the Control Center master host.
4. Log in to the Control Center master host as root, or as a user with superuser privileges.
5. Stage the files in `/tmp`.
 - a. Create a directory in `/tmp` for the files.
The directory must be local (not mounted) and must be readable, writable, and executable by all users. For example, `/tmp/impact`.

```
mkdir /tmp/impact
```

- b. Copy or move the downloaded files to `/tmp/impact`.
- c. Change the file permissions.
The files must have the same permissions as their parent directory.

```
chmod -R 777 /tmp/impact
```

6. Install the image.
 - a. Change directory to the location where the image file is stored.
For example, `/tmp/impact`:

```
cd /tmp/impact
```

- b. Install the image.

```
yes | ./install-zenoss-impact_5.3_5.3.4.0.1_34.run
```

Installing the export fix package

Perform this procedure to install export fix package prior to updating to 5.5.x.

1. Log in to the Control Center browser interface.
2. In the Application column of the Applications table, click Resource Manager.
3. Scroll down to the Services table, and then locate the Impact service in the Infrastructure section.
4. Click Impact, and then locate the State Change Queue Length graph.
5. Log in to the Control Center master host as a user with serviced CLI privileges.
6. Stop the zenimpactstate service, and then verify that it stopped.

- a. Stop the zenimpactstate service.

```
serviced service stop zenimpactstate
```

- b. Verify that the service is stopped.

```
watch serviced service status zenimpactstate
```

7. In the Control Center browser interface, monitor the length of the state change queue.
When the queue length is 0 (zero), proceed to the next step.
8. Stop the Infrastructure/Impact service, and then verify that it stopped.

- a. Stop the Infrastructure/Impact service.

```
serviced service stop Infrastructure/Impact
```

- b. Verify that the service is stopped.

```
watch serviced service status Infrastructure/Impact
```

9. Extract the upgrade script from the ZenPacks.zenoss.ImpactServer ZenPack, make the script executable, and start the upgrade script.

- a. Change to the directory in which the Service Impact ZenPack egg file is located.
For example, the /tmp/impact directory:

```
cd /tmp/impact
```

- b. Extract the upgrade script from the ZenPacks.zenoss.ImpactServer egg file.
Replace Version with the ZenPack version number:

```
unzip -p ZenPacks.zenoss.ImpactServer-5.3.4.0.1-75-py2.7.egg ZenPacks/zenoss/ImpactServer/upgrade  
/upgrade.sh > upgrade.sh
```

- c. Make the script executable.

```
chmod +x upgrade.sh
```

- d. Start the upgrade script.

```
./upgrade.sh
```

Note: The upgrade script might display CRITICAL warning messages, which can be ignored.

10. Start the Infrastructure/Impact service, and then verify that it started.

- a. Start the Infrastructure/Impact service.

```
serviced service start Infrastructure/Impact
```

- b. Verify that the service is started.

```
watch serviced service status Infrastructure/Impact
```

11. Log in to the Control Center browser interface.

12. Restart Zenoss services.

```
serviced service restart Zenoss.resmgr/Zenoss
```

Alternatively, restart the Zenoss services by using the Control Center browser interface.

Downloading and staging 5.5.x packages

To perform this procedure, you need:

- A workstation with internet access.
- Permission to download files from delivery.zenoss.com. To request permission, file a ticket at the [Zenoss Support](#) site.
- A secure network copy program.

Follow these steps:

1. In a web browser, navigate to the download site, and then log in.
The download site is delivery.zenoss.com.
2. Download the Docker image file for Service Impact.
 - `install-zenoss-impact_VERSION.run`Replace VERSION with the most recent version number available on the download page.
3. Download the Service Impact ZenPack files.
 - `ZenPacks.zenoss.ImpactServer-VERSION-py2.7.egg`
 - `ZenPacks.zenoss.Impact-VERSION-py2.7.egg`Replace VERSION with the most recent version number available on the download page.
4. Use a secure copy program to copy the Docker image file and the ZenPack files to the Control Center master host.

Do not change the file names during the copy.

5. Log in to the Control Center master host as root, or as a user with superuser privileges.
6. Stage the Docker image file and ZenPack files in `/tmp`.
 - a. Create a directory in `/tmp` for the files.
The directory must be local (not mounted) and must be readable, writable, and executable by all users. For example, `/tmp/impact`.

```
mkdir /tmp/impact
```

- b. Copy or move the Docker image file and ZenPack files to `/tmp/impact`.
- c. Change the file permissions.
The files must have the same permissions as their parent directory.

```
chmod -R 777 /tmp/impact
```

Installing Service Impact 5.5.x packages

Before performing this procedure, remove the previous version. For more information, see [Installing or updating Service Impact](#).

Perform these steps:

1. Log in to the Control Center master host as root or as a user with superuser privileges.
2. Rename the Service Impact database file.
 - a. Display the Control Center tenant identifier.

```
ls /opt/serviced/var/volumes
```

The result should be one 24-character string. If the result is more than one string, please contact Zenoss Support.

- b. Rename the database file.
Replace Tenant-ID with the result from the previous command.

```
mv /opt/serviced/var/volumes/Tenant-ID/impact/db /opt/serviced/var/volumes/Tenant-ID/impact/db.pre-5.5.x
```

3. Install the new Service Impact image.
 - a. Change to the directory in which the Service Impact image is located.

```
cd /tmp/impact
```

- b. Install the image.

```
yes | ./install-zenoss-impact_*.run
```

4. Install the ZenPacks.zenoss.ImpactServer ZenPack.
Replace *VERSION* with the ZenPack version number:

```
serviced service run zope zenpack-manager install ZenPacks.zenoss.ImpactServer-VERSION-py2.7.egg
```

5. Start the Impact service, and then verify that it started.
 - a. Start the Impact service.

```
serviced service start Infrastructure/Impact
```

- b. Verify that the service is started.

```
watch serviced service status Infrastructure/Impact
```

6. Install the ZenPacks.zenoss.Impact ZenPack.
Replace *VERSION* with the ZenPack version number:

```
serviced service run zope zenpack-manager install ZenPacks.zenoss.Impact-VERSION-py2.7.egg
```

7. Identify the delegate hosts where Resource Manager is running.

```
serviced host list --show-fields=Pool,Name
```

The name of the resource pool where Resource Manager is running is not standardized.

8. Log in to a delegate host, and then start a Zope container.
 - a. Log in to a delegate host in the Resource Manager pool as root or as a user with `serviced` CLI privileges.
 - b. Start a Zope container on the host.

```
serviced service shell zope
```

- c. In the container, log in as user zenoss.

```
su - zenoss
```

9. Update the catalog and then initialize the server.
 - a. Update the catalog.

```
zenimpactgraph run -x catclean
```

- b. Initialize the Service Impact server (Neo4j) database.

```
zenimpactgraph run --update
```

Processing may appear stuck for tens of minutes as connected objects are found in the ZODB. This is normal.

- c. Exit the zenoss user account.

```
exit
```

- d. Exit the Zope container.

```
exit
```

10. Rename dynamic services and organizers, if necessary.

```
zenimpactgraph run -x change_nodes_ids
```

The script checks for characters that are not allowed in names and replaces them. Changes are reported as they are made.

11. Restart all Zenoss services:

```
serviced service restart Zenoss.resmgr/Zenoss
```

12. Refresh the reverse proxy cache:

```
serviced service restart Zenoss.resmgr --auto-launch=False
```

13. Refresh the cache in your browser. The procedure varies by browser.

Integrating Service Impact and Resource Manager

Before performing this procedure, perform all of the previous update procedures. For more information, see [Installing or updating Service Impact](#).

1. Identify the delegate host where the zenimpactstate container is running, and the ID of the container.
 - a. Log in to the Control Center master host as a user with serviced CLI privileges.
 - b. Display the hostname where the container is running.

```
serviced service status zenimpactstate --show-fields=Name,Hostname
```

2. On your workstation, copy [the GraphML file exported previously](#) to a temporary directory on the delegate host identified in step 1.
3. Log in to the delegate host as root or as a user with superuser privileges.
4. Display the ID of the container in which the zenimpactstate service is running.

```
docker ps -aqf "name=$(serviced service list | awk '/zenimpactstate/ { print $2}')"
```

5. Copy the GraphML file from the delegate host to a directory inside the zenimpactstate container.
 - a. Copy the file to the /tmp directory inside the container.
Replace MyExportFile with the name of your GraphML file, and replace ContainerID with the result of the docker command in step 4:

```
docker cp /tmp/MyExportFile.graphml ContainerID:/tmp
```

- b. Log in to the zenimpactstate container as root.

```
serviced service attach zenimpactstate
```

- c. Move the GraphML file to the zenoss user HOME directory and change its owner and group.

```
mv /tmp/MyExportFile.graphml /home/zenoss  
chown zenoss:zenoss /home/zenoss/MyExportFile.graphml
```

- d. Switch user to zenoss.

```
su - zenoss
```

6. Import of the GraphML file into Service Impact.
For detailed information about importing the file, see [Importing service model definitions](#). The following substeps provide an outline of the procedures.

- a. Initiate the import.

```
zenimpactimport ./MyExportFile.graphml | tee reconcile.file
```

- b. Reconcile entities as necessary.
For more information, see [Reconciling originating and target system entities](#).
- c. Commit the reconciled entities.

```
zenimpactimport -r ./reconcile.file --commit ./MyExportFile.graphml 2>&1 | tee error.out
```

Updating both Service Impact ZenPacks

Before performing this procedure, complete prerequisites listed in [Installing or updating Service Impact](#).

Perform this procedure to update **both** Zenpacks, ZenPacks.zenoss.Impact and ZenPacks.zenoss.ImpactServer.

1. Log in to the Control Center browser interface.
2. In the Application column of the Applications table, click Resource Manager.
3. Scroll down to the Services table, and then locate the Impact service in the Infrastructure section.
4. Click Impact, and then locate the State Change Queue Length graph.
5. Log in to the Control Center master host as a user with serviced CLI privileges.
6. Stop the zenimpactstate service, and then verify that it stopped.
 - a. Stop the zenimpactstate service.

```
serviced service stop zenimpactstate
```

- b. Verify that the service is stopped.

```
watch serviced service status zenimpactstate
```

7. In the Control Center browser interface, monitor the length of the state change queue. When the queue length is 0 (zero), proceed to the next step.
8. Stop the Infrastructure/Impact service, and then verify that it stopped.
 - a. Stop the Infrastructure/Impact service.

```
serviced service stop Infrastructure/Impact
```

- b. Verify that the service is stopped.

```
watch serviced service status Infrastructure/Impact
```

9. Extract the upgrade script from the ZenPacks.zenoss.ImpactServer ZenPack, make the script executable, and start the upgrade script.
 - a. Change to the directory in which the Service Impact ZenPack egg file is located. For example, the /tmp/impact directory:

```
cd /tmp/impact
```

- b. Extract the upgrade script from the ZenPacks.zenoss.ImpactServer egg file. Replace VERSION with the ZenPack version number:

```
unzip -p ZenPacks.zenoss.ImpactServer-VERSION-py2.7.egg ZenPacks/zenoss/ImpactServer/upgrade/upgrade.sh > upgrade.sh
```

- c. Make the script executable.

```
chmod +x upgrade.sh
```

- d. Start the upgrade script.

```
./upgrade.sh
```

Note: The upgrade script might display CRITICAL warning messages, which can be ignored.

10. Start the Infrastructure/Impact service, and then verify that it started.
 - a. Start the Infrastructure/Impact service.

```
serviced service start Infrastructure/Impact
```

- b. Verify that the service is started.

```
watch serviced service status Infrastructure/Impact
```

11. Install the ZenPacks.zenoss.Impact ZenPack. Replace VERSION with the ZenPack version number:

```
serviced service run zopec zenpack-manager install ZenPacks.zenoss.Impact-VERSION-py2.7.egg
```


12. Restart Zenoss services.

```
serviced service restart Zenoss.resmgr/Zenoss
```

Alternatively, restart the Zenoss services by using the Control Center browser interface.

Rebuilding the graph database

Use this procedure after updating Service Impact from version 5.4.x to 5.5.0. If you are updating from a previous release, use a different procedure. For more information, see [Installing or updating Service Impact](#).

The rebuild process can take a long time. Perform this procedure during a maintenance window.

Perform these steps:

1. Log in to the Control Center master host as root or as a user with superuser privileges.
2. Stop the Zenoss services, and then verify that the services are stopped.
 - a. Stop Resource Manager.

```
serviced service stop Zenoss.resmgr/Zenoss
```

- b. Wait until all services are stopped.
Use the `watch` command to monitor the status.

```
watch serviced service status Zenoss.resmgr/Zenoss
```

3. Rename the Service Impact database file.
 - a. Stop the Impact server.

```
serviced service stop Infrastructure/Impact
```

- b. Wait until the service is stopped.

```
serviced service status Infrastructure/Impact
```

- c. Display the Control Center tenant identifier.

```
ls /opt/serviced/var/volumes
```

The result should be one 24-character string. If the result is more than one string, please contact Zenoss Support.

- d. Rename the database file.
Replace Tenant-ID with the result from the previous command.

```
mv /opt/serviced/var/volumes/Tenant-ID/impact/db /opt/serviced/var/volumes/Tenant-ID/impact/db.pre-5.5.x
```

- e. Start the Impact server.

```
serviced service start Infrastructure/Impact
```

- f. Wait until the service is started.

```
serviced service status Infrastructure/Impact
```

4. Start the Zenoss service that is needed to perform the graph rebuild.
 - a. Start the `zeneventserver` service.

```
serviced service start zeneventserver
```

- b. Wait until the service is started.
Use the `watch` command to monitor the status.

```
watch serviced service status zeneventserver
```

5. Identify the delegate hosts where Resource Manager is running.

```
serviced host list --show-fields=Pool,Name
```

The name of the resource pool where Resource Manager is running is not standardized.

6. Log in to a delegate host, and then start a Zope container.

The rebuild process requires approximately 1GB to 2GB per CPU core. If the delegate host does not have adequate main memory, the host may become unstable.

- a. Log in to a delegate host in the Resource Manager pool as root or as a user with `serviced` CLI privileges.
- b. Start a Zope container on the host.

```
serviced service shell zope
```

- c. In the container, log in as user zenoss.

```
su - zenoss
```

7. Update the catalog and then initialize the server.

- a. Update the catalog.

```
zenimpactgraph run -x catclean
```

- b. Update the Service Impact graph in ZODB and [Neo4j](#).

```
zenimpactgraph run --update
```

For more information about the update process, see [Graph update tips](#).

- c. Exit the zenoss user account.

```
exit
```

- d. Exit the Zope container.

```
exit
```

8. Restart all Zenoss services:

```
serviced service restart Zenoss.resmgr/Zenoss
```

9. Refresh the cache in your browser. The procedure varies by browser.

Graph update tips

To maintain consistency with the Resource Manager model database, update the Service Impact graph database when most Resource Manager services (except Infrastructure services) are stopped. Typically, this requires a maintenance window. For more information, see [Rebuilding the graph database](#).

During a graph update, the `zenimpactgraph` script creates one worker process for each available CPU core. Each worker requires up to 2GB of main memory. Without it, the host where the update is running may become unstable.

Processing may appear stuck for tens of minutes as connected objects are found in the ZODB and their graphs are built. For complex objects, the build process can take a couple of minutes each. Status is reported every 200 nodes, so processing can take a long time.

The graph update process proceeds as follows:

1. Scan the model database for edges.
2. Add edges to the graph database.
3. Color the dynamic service tree.
4. Push the tree to impact-server.

If the process fails during steps 3 or 4, you can resume the process with the following command:

```
zenimpactgraph run -x color -x push
```

To determine whether steps 1 and 2 have completed successfully, look for the following line in the output:

```
0 edges failed to be added to impact graph
```

The output of the graph update process may include a warning similar to the following example:

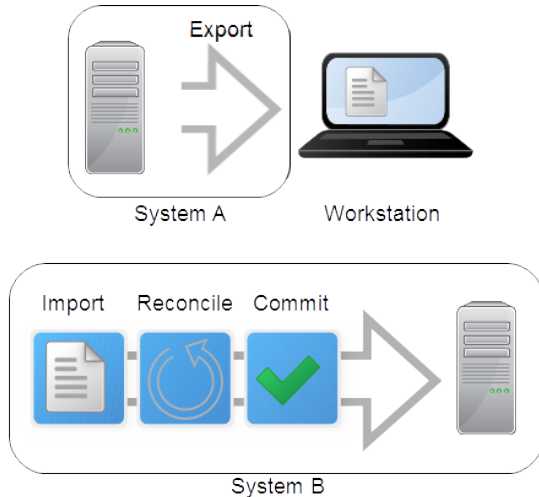
```
WARNING zen.ImpactGraph: Invalid edge:source node not available for Provider: ZenPacks.zenoss.DynamicView.model.adapters.DeviceOrganizerRelationsProvider; Source: d92e8547-15bf-45b0-8688-ca6f2576593e; Target: 8608ce82-f627-44a2-8bb4-692c8e58a339 - cannot find object for d92e8547-15bf-45b0-8688-ca6f2576593e
```

The typical cause is an inconsistent ZODB catalog. If the missing edges are important to your service models, the catalog may need to be rebuilt. For assistance, please contact Zenoss Support.

Exporting service models from one system to another

You can create a service model on one Resource Manager system (the originating system) and export that model to another system (the target system). This feature is useful for tasks such as repopulating a test or staging environment from a production environment, and performing repeated exporting and model synchronization from develop, to staging, to production environments. Review the following use case example and process overview.

"Project Zebra" is a complicated new E-commerce web application with many clustered entities. The application has been tested in a staging environment (the originating Resource Manager system), and the service model is ready to export from staging to the production environment (the target Resource Manager system). The following diagram illustrates the steps in the process; System A is the originating system and System B is the target system.



Exporting

On the originating system, select the dynamic service, dynamic service organizer, or set of dynamic services to export their service models. The export process generates a file in GraphML XML format that describes the structural properties of the service models, including the devices, components, logical nodes, and their relationships. On your workstation, save the exported file (referred to in this document as `svc_export.graphml`).

The export file contains definitions for entities that are explicitly defined in the service model. It does not include nodes that Service Impact identified as having an impact relationship and added. For example, service model SM1 defines only an application server device member, App Server A. However, Service Impact evaluated App Server A and infrastructure dependencies on the VM, VM OS, VM host, and file systems on which it resides. Service Impact automatically added the related service model members to the service model SM1. When service model SM1 is exported, the export file contains only service model SM1 and App Server A. When service model SM1 is imported and committed on the target system, a new version of service model SM1 is generated that is unique to the infrastructure dependencies of the target system.

Importing

On the target system, you import the exported file, which imports the service model definitions to view into a working "sandbox" testing environment. While in the sandbox, service models are not operational and are not available in the Resource Manager browser interface. Before moving a service model out of the sandbox, you must complete the reconcile and commit steps.

The initial import process automatically attempts to reconcile (match) each service model member that the exported file references with a corresponding entity on the target system. For example, for Project Zebra, some devices in the staging and production environments might be shared. Others, like the database, will be replaced in the production environment by different entities in the same role. With the higher scalability that production requires, the clusters in the production environment might have more entities. Some entities might not exist or be known on the target system. Such differences must be reconciled before the new application is moved into production.

The import process generates text files to report the results and identify UNRECONCILED entities. This document refers to the files as `svc_export.graphml.latest.txt` and `svc_export.graphml.nnnn.txt`.

Reconciling

The generated text files provide known details about UNRECONCILED entities. Using that information, you search for and edit `svc_export.graphml.latest.txt` to identify corresponding entities on the target system. Your edited version of `svc_export.graphml.latest.txt` provides input for each reconcile attempt.

Committing

After you have reconciled the service models in the sandbox, perform the final commit. The committed service models are active and begin analyzing Resource Manager events when appropriate, and generating service events. View and work with the service models in the SERVICES tab of the Resource Manager browser interface.

Details about entities that required manual reconciliation are retained on the target system. Future service model imports of the same entities will be reconciled automatically. For example, for Project Zebra, the first import-reconcile-commit process on System B required a manual mapping of database "Test_DB1" to "Prodxn_DB1." Future imports on System B of service models that reference "Test_DB1" will automatically refer to "Prodxn_DB1" instead.

You cannot reverse a commit; however, you can start a new import process or edit the service model definition in the Resource Manager browser interface.

Exporting a service model

Select service models on the originating system and generate the export file that you will use as input to import on the target system.

1. In the Resource Manager browser interface, select the SERVICES tab.
2. In the tree view, select the service models to export.
3. From the Action menu at the bottom of the tree view, select Export Selected.
4. Save the generated export file (svc_export.graphml) on your workstation.
5. Proceed with [Importing service model definitions](#).

Importing service model definitions

Perform the procedures in this section to import a service export file in graphml format that contains service model definitions.

The service models in the export file are imported into a non-operational "sandbox" environment on the target system. The service model remains hidden in the sandbox through the reconcile phase, and does not appear in the browser interface until you perform the final commit.

The import and reconcile process typically supports files that were exported by the same or different Resource Manager instance and version (older or newer), and Service Impact versions (older or newer). If an export file's content is incompatible with a version of import, incompatibilities can usually be resolved by adjusting the graphml file content or structure.

- [Preparing to import](#)
- [Initiating the import](#)
- [Reconciling originating and target system entities](#)
- [Attempting to reconcile](#)
- [Canceling the service model reconcile](#)
- [Adding the service models to the target system](#)
- [Future export scenarios and machine learning](#)
- [zenimpactimport](#)
- [Example export file](#)
- [Example import results file](#)

Preparing to import

Before you can import, you must copy the export file from your workstation to the target system master host and connect to that master host.

Prerequisite: Generate an export file (see [Exporting a service model](#)).

1. Log in to the Control Center host on which the zenimpactstate service is running.
2. Create a directory for the exported file on the Control Center master host.
The directory that contains the exported file must be a local directory (not mounted) and must be readable, writable, and executable by all users.
The following example creates a directory in /tmp:

```
mkdir /tmp/impact && chmod 777 /tmp/impact
```

3. Change directory to the directory you created.
For example:

```
cd /tmp/impact
```

4. Log in to the Control Center master host as a user with serviced CLI privileges.
5. Copy the exported graphml file from your workstation to the new directory (if on the same host) or use scp to perform a network copy.
For example:

```
scp export_impact_svcname_20161215203455.graphml zenossuser@impacthost.company.com
```

6. Connect to the zenimpactstate container.

```
serviced service shell -i zenimpactstate
```

7. In the new session, switch user from root to zenoss.

```
su - zenoss
```

8. Change directory to the pwd directory:
For example:

```
cd /mnt/pwd
```

Proceed with the initial import.

Initiating the import

In the initial import, service models from the exported file are imported into the sandbox, the product attempts to reconcile service model members from the originating system with entities on the target system, and output files are generated to show the result of the reconciliation attempt and actions to take. Prerequisites:

- Generate an export file (see [Exporting a service model](#)).
- Copy the export file to the target system and log in (see [Preparing to import](#)).

1. Initiate the import.

```
zenimpactimport -i svc_export.graphml
```

The exported data is imported to the target system; the first reconciliation attempt is made, and generated text files report the results.

2. Review import results in file `svc_export.graphml.latest.txt`, and then proceed as follows:
- If Nodes unreconciled is 0, proceed with [Adding the service models to the target system](#).
 - If Nodes unreconciled is 1 or greater, proceed with [Reconciling originating and target system entities](#).
 - If the import ends in an error or you do not plan to commit the import, proceed with [Canceling the service model reconcile](#).

Reconciling originating and target system entities

The reconcile phase is an iterative process in which you attempt to match devices or components from the originating system with their equivalents on the target system.

Multiple attempts to reconcile are often required before you can commit a service model that originated on another system. For faster progress, make multiple small changes and frequent attempts to reconcile.

Begin by reviewing the text files that the initial import generated. Files `svc_export.graphml.latest.txt` and `svc_export.graphml.0001.txt` are identical. Each attempt to reconcile generates a new "latest" file and a new, sequentially numbered file to provide a record of your changes. Do not modify the numbered files.

For each service model member, one of the following results or actions exists. When the attempt to reconcile succeeds, results are reported. When you must manually correct exceptions in the model, actions provide guidance. `<NodeID>` identifies the imported service model member. Do not modify the node ID. `<TargetNodeID>` is the DMD ID, name, or GUID of the target entity to apply to the imported service model member.

UNRECONCILED `<NodeID>`

The result for any member on the originating system that the service model references but which cannot be identified on the target system.

MAP `<NodeID>` `<TargetNodeID>`

The result if a matching entity was found on the target system or an action to take for an UNRECONCILED service model member. Replace UNRECONCILED with MAP and specify the DMD ID, name, or GUID for the target entity. For example, you might MAP to specify that device "stage-AppSvr1" should reference "prod-AppSvr1" in production.

```
MAP stage-AppSvr1 prod-AppSvr1
```

Note: Some cases might require that you first define and model a device in Resource Manager so that the next reconcile attempt can detect it either automatically or manually through use of the MAP action.

CREATE `<NodeID>`

An action to take when no matching entity was found in the target system, but an entity can be created through import. For example, you might CREATE to add a device for production when a similar device does not exist in staging.

```
CREATE prod-AppSvr1
```

DELETE `<NodeID>`

The result when a matching entity was found in the target system, and it is marked to be deleted in the import file, or an action to take when an entity is not needed. Use DELETE to delete the device, not just remove it from the service model. Replace UNRECONCILED with DELETE and specify the DMD ID, name, or GUID.

For example, you might DELETE to remove a device in production.

```
DELETE prod-AppSvr1
```

If the `<NodeID>` does not exist in the target system, the action is converted to IGNORE.

IGNORE `<NodeID>`

An action to take for an imported service model member. For example, you might IGNORE because a similar entity does not exist in production.

```
IGNORE stage-AppSvr1
```

Before you commit, you must edit `svc_export.graphml.latest.txt` (or another version of the file) to specify action data. That is, replace UNRECONCILED entries with MAP, CREATE, DELETE, or IGNORE action syntax, and then attempt to reconcile.

Note: In some cases, such as when target devices are not yet online, you might commit and make the service model active even with a few UNRECONCILED entries. Reconcile the remaining UNRECONCILED entries later by importing the same export file or a new file. This approach is useful if you plan to repeat the export to the same target, such as after upgrades or service model changes. You will not need to manually reconcile the same entries each time. You can also manually adjust committed service model definitions later in the target system by using the browser interface.

Attempting to reconcile

Prerequisites:

- Generate an export file (see [Exporting a service model](#)).
- Copy the export file to the target system and log in (see [Preparing to import](#)).
- Complete the initial import (see [Initiating the import](#)).
- Review [Reconciling originating and target system entities](#).

For more information about the `zenimport` command and options, see [zenimport](#) or access the help information:

```
zenimport -h
```

1. In `svc_export.graphml.latest.txt` or another version of the file, replace UNRECONCILED with an action to specify how the exception should be reconciled.
2. Change directory to the directory that contains files `svc_export.graphml.latest.txt` and `svc_export.graphml.nnnn.txt`.
3. Initiate the reconcile attempt.

```
zenimport -r svc_export.graphml
```

The `zenimport` command attempts to reconcile the originating and target systems, and generates new versions of files `svc_export.graphml.latest.txt` and `svc_export.graphml.nnnn.txt`.

4. Review import results in file `svc_export.graphml.latest.txt`, and proceed as follows:
 - If Nodes unreconciled is 1 or greater, repeat the reconciliation process.
 - If Nodes unreconciled is 0, proceed with [Adding the service models to the target system](#).
 - If the import ends in an error or you do not plan to commit the import, proceed with [Canceling the service model reconcile](#).

Canceling the service model reconcile

You might need to remove an uncommitted import attempt from the target system sandbox and free up sandbox resources.

Prerequisites:

- Generate an export file (see [Exporting a service model](#)).
- Copy the export file to the target system and log in (see [Preparing to import](#)).
- Complete the initial import (see [Initiating the import](#)).

You can specify the import attempt to cancel in the following ways:

By import file name:

```
zenimpactimport --import filename.graphml --abort
```

By reconcile file name:

```
zenimpactimport --reconcileFile filename.graphml.latest.txt --abort
```

By action file name:

```
zenimpactimport --actionFile filename.graphml*.latest.txt* --abort
```

By source and version. First, generate a list of import attempts:

```
zenimpactimport --list-imports
```

Then specify source, version, or both:

```
zenimpactimport --source sourceID --abort
```

```
zenimpactimport --import-version versionID --abort
```

```
zenimpactimport --source sourceID --import-version versionID --abort
```

The following example lists the import versions, the command to remove an uncommitted import, and the resulting list of import versions.

```
$ zenimpactimport --list-imports
Date                Source                Version                Status
2018-02-13 14:59:36 d6fc-b4fe-00505694381d 1481662776493 COMMITTED
2018-02-14 13:50:48 bd6d-a34a-0242ac110013 1481745048516 COMMITTED
2018-02-14 14:22:11 bd70-945b-0242ac11003d 1481746931561 UNCOMMITTED
2018-02-14 14:46:43 bd70-945b-0242ac11003d 1481748403174 UNCOMMITTED

$ zenimpactimport -r svc_export.graphml --import -version 1481746931561 --abort
INFO:zen.impact.import:Reading action data from svc_export.graphml.latest.txt
INFO:zen.impact.import:Aborting import bd70-945b-0242ac11003d 1481746931561 ...
INFO:zen.impact.import:Printing report to svc_export.graphml.0003.txt
INFO:zen.impact.import:Printing report to svc_export.graphml.latest.txt
INFO:zen.impact.import:Nodes unreconciled: 0
INFO:zen.impact.import:Import node status: aborted
INFO:zen.impact.import:Done

$ zenimpactimport --list-imports
Date                Source                Version                Status
2018-02-13 14:59:36 d6fc-b4fe-00505694381d 1481662776493 COMMITTED
2018-02-14 13:50:48 bd6d-a34a-0242ac110013 1481745048516 COMMITTED
2018-02-14 14:46:43 bd70-945b-0242ac11003d 1481748403174 UNCOMMITTED
```

Adding the service models to the target system

When the "latest" file identifies no UNRECONCILED nodes, commit the reconciled service models that are in the sandbox. After the commit, imported service models, organizers, logical nodes, global and contextual policies, and custom state providers are added to the target system. Service models are operationally active to process device and service events, and are listed in the SERVICES tab of the browser interface.

Prerequisites:

- Generate an export file (see [Exporting a service model](#)).
- Copy the export file and log in to the target system (see [Preparing to import](#)).
- Complete the initial import (see [Initiating the import](#)).
- Reconcile devices or components from the originating system with their equivalents in the target system (see [Attempting to reconcile](#)).
- Review the following considerations:
 - It is possible to commit the file with UNRECONCILED nodes, however, those members are lost from the resulting service model.
 - A final commit cannot be reversed. To change the model definition after committing, edit the service model definition in the SERVICES tab or delete the service model and start a new export process.

You can specify the import attempt to commit in the following ways:

By import file name:

```
zenimpactimport --import filename.graphml --commit
```

By reconcile file name:

```
zenimpactimport --reconcileFile filename.graphml.latest.txt --commit
```

By action file name:

```
zenimpactimport --actionFile filename.graphml*.latest.txt* --commit
```

By source and version. First, generate a list of import attempts:

```
zenimpactimport --list-imports
```

Then specify source, version, or both:

```
zenimpactimport --source sourceID --commit
```

```
zenimpactimport --import-version versionID --commit
```

```
zenimpactimport --source sourceID --import-version versionID --commit
```

Future export scenarios and machine learning

In the future, service models for dynamic services that you import might reference the same service model members that required manual reconcile actions in a previous import. The previous manual reconciliation and machine learning of mapped members simplifies each future export.

For example, the service model for dynamic service "S" in a Resource Manager staging environment refers to devices A,B, and C. In the production environment, the imported service model must be reconciled to map to the shared device A, and to equivalent devices X and Y instead of B and C.

Initial import of service model "S" into production

After the initial import, B and C were UNRECONCILED. You created MAP actions to reconcile B to X and C to Y, and then committed with `svc_export.graphml.latest.txt`.

Import 2 of service model "S" to the same production system

A few weeks later, an upgrade to service model "S" adds device "D." The changed service model "S" with devices A,B, C, D is imported into production, where service model "S" must have entities A,X,Y, Z. Previously, you mapped and committed the service model with B mapped to X and C mapped to Y. After import 2, file `svc_export.graphml.latest.txt` shows only one UNRECONCILED entry, for added device "D." Resource Manager learned and automatically mapped B and C to X and Y. This time, you only need to manually add a MAP action for "D" to "Z", and then commit.

Import 3: initial import of the service model for dynamic service "T"

A third import occurs into production for a different service, "T." Service model "T" contains devices A and D. After the import and automatic reconcile attempt, no UNRECONCILED entries exist for "T." That is, Resource Manager automatically found and mapped A, and used knowledge from your previous mapping of device D to Z for a different service import. In production, service model "T" contain devices A and Z.

Import 4: service model "T" with a different mapping of an entity than in "S" service context

A fourth import occurs, this time of service model "T" with service model members A, B, D. However, in the context of service "T," device B should be mapped to device "P." In production, service model "T" should contain service model members A, P, Z.

After the import and automatic reconcile attempt, no UNRECONCILED entries exist, but `svc_export.graphml.latest.txt` shows D mapped to Z and B mapped X. You must manually MAP B to P, reconcile, and then commit. As a result, the model for service "T" contains A, P, Z.

zenimpactimport

DESCRIPTION

zenimpactimport - Imports a [GraphML](#) file that contains Service Impact the service model definitions for a dynamic service; imports multiple subsequent calls to process new actions in order to reconcile a service model against the entities in Resource Manager.

The import process generates files that report the result of the import and recommend actions. The files follow the same format as the --action command line parameter. An action consists of an action keyword followed by parameters for that type of action. Actions and parameters are separated by white space, with each action on a separate line.

For more information about the import process, see [Importing service model definitions](#).

SYNTAX

```
zenimpactimport (IMPORT_FILE | IMPORT_ID | ACTION_FILE) ACTIONS* [TRANSACTION]
```

COMMANDS and OPTIONS

- import, -i
Perform the import operation.
- reconcile, -r
Perform the reconcile operation.
- help, -h
Show the help information.
- list-imports
Displays a list of existing imports and their status. Each import is uniquely identified by source and version identifiers.
- list-adapters
Displays a list of adapters that are installed to parse and recognize export file content and structure. The "zenoss" adapter expects a Zenoss Service Impact service export file in graphml format.

IMPORT_FILE

--import IMPORT_FILENAME [--adapter ADAPTER]

Performs the initial import of a file in graphml format that generates and attempt to reconcile service model members with entities on the target Resource Manager system.

IMPORT_FILENAME

The file that contains the data in a format that the ADAPTER recognizes. The default file name is svc_export.graphml.

ADAPTER

The name of an input adapter. Currently, only the default adapter for the Zenoss graphml export format is supported.

IMPORT_ID

--source SOURCEID --import-version VERSIONID

Performs the import of the attempt that you specify by source ID, version ID, or both. Each import is uniquely identified by source and version identifiers. To display imports, issue the --list-imports command.

The *.latest.txt file is rewritten by the latest command that uses the specified combination of *sourceID* and *versionID*.

ACTION_FILE

(--actionFile | --reconcileFile) FILENAME

Performs the import using data in the specified file.

FILENAME

A text file that contains reconcile actions. If no FILENAME is specified on the command line, file svc_export.graphml.file.latest.txt is used as input.

ACTIONS

--action ACTION

Provides a way to execute a single reconciliation action on the command line, rather than editing the input file or the parameter --actionFile. For more information about the actions, see [Reconciling originating and target system entities](#).

CREATE NodeID

Creates the specified node on the target system.

DELETE NodeID

Deletes the specified node from the target system.

MAP NodeID TargetNodeID

Replaces the specified node on the originating system with the specified node on the target system.

IGNORE NodeID

Takes no action for the specified node.

UNRESOLVED NodeID

The specified node is referenced in the service model but it cannot be reconciled with a node on the target system.

TRANSACTION

{--abort|--commit}

Abort or commit the import attempt that is identified by its unique combination of source ID and version ID. Aborting destroys the import sandbox. Committing makes all entries in the input file permanent and operational in the target system. The last committed combination is the final result.

Example export file

The generated export file name is in the format `export_impact_svcname_YYYYMMDDHHMMSS.graphml`, where

- `export` is the name of the action that is performed.
- `impact` is the name of the Resource Manager plugin that performs the action.
- `service` is the name of the service model, a service organizer, or dynamic services environment that you selected for export.
- `YYYYMMDDHHMMSS` is the timestamp of the export action.

For example, `export_impact_DynamicServices_20160331135219.graphml`. Note: This document refers to the file as `svc_export.graphml`. The export file name is included in the names of the text files that the import process and each reconciliation attempt generate. For example, if file `export_impact_DynamicServices_20160331135219.graphml` is imported, then the text file names are:

`export_impact_DynamicServices_20160331135219.graphml.latest.txt`

`export_impact_DynamicServices_20160331135219.graphml.nnnn.txt`

The following example shows the export file for a service named `BizSvcA`. The service model contains types `DEVICE`, `SERVICE`, and `COMPONENT` (`PROP_element_type_id`).

- The service is named `BizSvcA` (`PROP_name`) and it is a dynamic **service** (`PROP_meta_type`).
- The devices are named `PROD_SQL_SERVER` and `vcloud-01.zenosslabs.com` (`PROP_name`). They are a device and a `vCloudCell`, respectively (`PROP_meta_type`).
- The components are named `APP_DB1` and `tempdb` (`PROP_name`). They are a `WinSQLDatabases` (`PROP_meta_type`).
- Four impact relationship edge members from each device and component.

```
<?xml version="1.0" encoding="utf-8"?>
<graphml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns http://graphml.graphdrawing.org/xmlns/1.1/graphml.xsd">
  <key attr.name="PROP_production" attr.type="string" for="node"/>
  <key attr.name="INTRINSIC_STATE_AVAILABILITY" attr.type="string" for="node"/>
  <key attr.name="PROP_priority" attr.type="string" for="node"/>
  <key attr.name="INTRINSIC_STATE_PERFORMANCE" attr.type="string" for="node"/>
  <key attr.name="RELATED_EVENT_IDS" attr.type="string" for="node"/>
  <key attr.name="NODE_TYPE" attr.type="string" for="node"/>
  <key attr.name="DERIVED_STATE_PERFORMANCE" attr.type="string" for="node"/>
  <key attr.name="IN_ANY_CONTEXT" attr.type="string" for="node"/>
  <key attr.name="PROP_element_type_id" attr.type="string" for="node"/>
  <key attr.name="DERIVED_STATE_AVAILABILITY" attr.type="string" for="node"/>
  <key attr.name="ID" attr.type="string" for="node"/>
  <key attr.name="PROP_name" attr.type="string" for="node"/>
  <key attr.name="INTRINSIC_STATE_CAPACITY" attr.type="string" for="node"/>
  <key attr.name="PROP_meta_type" attr.type="string" for="node"/>
  <key attr.name="ORGANIZER" attr.type="string" for="node"/>
  <key attr.name="DESCRIPTION" attr.type="string" for="node"/>
  <key attr.name="LN_CRITERIA" attr.type="string" for="node"/>
  <key attr.name="LN_AVAILABILITY_MAP" attr.type="string" for="node"/>
  <key attr.name="LN_PERFORMANCE_MAP" attr.type="string" for="node"/>
  <key attr.name="CUSTOM_STATE_PROVIDER" attr.type="string" for="node"/>
  <key attr.name="LABEL" attr.type="string" for="edge"/>
  <graph edgedefault="directed">
    <node>
      <data key="PROP_production">1000</data>
      <data key="PROP_priority">3</data>
      <data key="NODE_TYPE">ELEMENT</data>
      <data key="DERIVED_STATE_PERFORMANCE">__DERIVED_STATE__|ACCEPTABLE|</data>
      <data key="IN_ANY_CONTEXT">true</data>
      <data key="PROP_element_type_id">DEVICE</data>
      <data key="DERIVED_STATE_AVAILABILITY">__DERIVED_STATE__|DOWN|</data>
      <data key="ID">8ddcc69b-e6a7-4370-8f3d-4a53f897b9f4</data>
      <data key="PROP_name">PROD_SQL Server</data>
      <data key="PROP_meta_type">Device</data>
      <data key="CUSTOM_STATE_PROVIDER">{}</data>
    </node>
    <node>
      <data key="PROP_production">1000</data>
      <data key="NODE_TYPE">ELEMENT</data>
      <data key="DERIVED_STATE_PERFORMANCE">__DERIVED_STATE__|ACCEPTABLE|</data>
      <data key="IN_ANY_CONTEXT">true</data>
      <data key="PROP_element_type_id">COMPONENT</data>
      <data key="DERIVED_STATE_AVAILABILITY">__DERIVED_STATE__|UP|</data>
      <data key="ID">fc6ab26d-75cd-491a-8c71-75fb25845f97</data>
    </node>
  </graph>
</graphml>
```

```

    <data key="PROP_name">APP_DB1</data>
    <data key="PROP_meta_type">WinSQLDatabase</data>
    <data key="CUSTOM_STATE_PROVIDER">{</data>
</node>
<node>
    <data key="PROP_production">1000</data>
    <data key="NODE_TYPE">ELEMENT</data>
    <data key="DERIVED_STATE_PERFORMANCE">__DERIVED_STATE__|ACCEPTABLE|</data>
    <data key="IN_ANY_CONTEXT">>true</data>
    <data key="PROP_element_type_id">COMPONENT</data>
    <data key="DERIVED_STATE_AVAILABILITY">__DERIVED_STATE__|UP|</data>
    <data key="ID">d66416a5-7db7-47df-b9ab-ed9422700efe</data>
    <data key="PROP_name">tempdb</data>
    <data key="PROP_meta_type">WinSQLDatabase</data>
    <data key="CUSTOM_STATE_PROVIDER">{</data>
</node>
<node>
    <data key="IN_ANY_CONTEXT">>true</data>
    <data key="PROP_element_type_id">SERVICE</data>
    <data key="DERIVED_STATE_AVAILABILITY">__DERIVED_STATE__|DOWN|</data>
    <data key="ID">0dcda951-9885-449c-af1b-0adb37aeec95</data>
    <data key="NODE_TYPE">SERVICE</data>
    <data key="PROP_name">BizSvcA</data>
    <data key="DERIVED_STATE_PERFORMANCE">__DERIVED_STATE__|ACCEPTABLE|</data>
    <data key="PROP_meta_type">DynamicService</data>
    <data key="ORGANIZER">/Zreconsvcs</data>
    <data key="DESCRIPTION"/>
</node>
<node>
    <data key="PROP_production">1000</data>
    <data key="PROP_priority">3</data>
    <data key="NODE_TYPE">ELEMENT</data>
    <data key="DERIVED_STATE_PERFORMANCE">__DERIVED_STATE__|ACCEPTABLE|</data>
    <data key="IN_ANY_CONTEXT">>true</data>
    <data key="PROP_element_type_id">DEVICE</data>
    <data key="DERIVED_STATE_AVAILABILITY">__DERIVED_STATE__|UP|</data>
    <data key="ID">6fca9724-6c92-4fd9-b8fa-c72bb133fda8</data>
    <data key="PROP_name">vcloud-01.zenosslabs.com</data>
    <data key="PROP_meta_type">vCloudCell</data>
    <data key="CUSTOM_STATE_PROVIDER">{</data>
</node>
<edge source="8ddcc69b-e6a7-4370-8f3d-4a53f897b9f4" target="0dcda951-9885-449c-af1b-0adb37aeec95">
    <data key="LABEL">IMPACTS</data>
</edge>
<edge source="fc6ab26d-75cd-491a-8c71-75fb25845f97" target="0dcda951-9885-449c-af1b-0adb37aeec95">
    <data key="LABEL">IMPACTS</data>
</edge>
<edge source="d66416a5-7db7-47df-b9ab-ed9422700efe" target="0dcda951-9885-449c-af1b-0adb37aeec95">
    <data key="LABEL">IMPACTS</data>
</edge>
<edge source="6fca9724-6c92-4fd9-b8fa-c72bb133fda8" target="0dcda951-9885-449c-af1b-0adb37aeec95">
    <data key="LABEL">IMPACTS</data>
</edge>
</graph>
</graphml>

```

Example import results file

The initial import generates `svc_export.graphml.latest.txt` to report the results of the process. The following example shows unreconciled, created, and mapped service model members.

Be aware that the `svc_export.graphml.latest.txt` is rewritten by the latest command that uses the specified combination of `sourceID` and `versionID`.

UNRECONCILED

In this example, the initial import could not match databases `DB3From`, `DB2From`, and `tempdbFrom` with databases on the target system. Manual reconciliation is needed. For UNRECONCILED entries, known information is displayed so that you can search the target system for a corresponding entity. In the example, `PROP_meta_type` indicates that a `WinSQLDatabase` is referenced, `PROP_element_type_id` shows that it is a `COMPONENT`, and `PROP_name` shows the database name. After the `EXTERNAL_ID` has been reconciled, subsequent outputs do not include these details.

CREATE

For imported service `DBSvcExpTarget`, a matching service does not exist on the target system. The import command generated a `CREATE` statement for that service and displays related details about the service that is being created.

MAP

For the SQL Server device and database from the originating system, the import process found and mapped the same members on the target system. Details about the members are not displayed because manual reconciliation is not needed.

```
$ cat DBSvcExpFROM.graphml.latest.txt
SOURCE      d96ce0f4-2d06-11e6-b939-0242ac110026
VERSION     1481917173580      # 2016-12-16 19:39:33

# CUSTOM_STATE_PROVIDER:      {}
# DERIVED_STATE_AVAILABILITY:  __DERIVED_STATE__|UP|
# DERIVED_STATE_PERFORMANCE:  __DERIVED_STATE__|ACCEPTABLE|
# EXTERNAL_ID:                d05e3670-0bbf-4e76-bcd2-a8a1cd333333
# ID:                          d05e3670-0bbf-4e76-bcd2-a8a1cd333333
# IN_ANY_CONTEXT:              true
# NODE_TYPE:                   ELEMENT
# PROP_element_type_id:        COMPONENT
# PROP_meta_type:              WinSQLDatabase
# PROP_name:                   DB3From
# PROP_production:             1000
UNRECONCILED    d05e3670-0bbf-4e76-bcd2-a8a1cd333333

# CUSTOM_STATE_PROVIDER:      {}
# DERIVED_STATE_AVAILABILITY:  __DERIVED_STATE__|UP|
# DERIVED_STATE_PERFORMANCE:  __DERIVED_STATE__|ACCEPTABLE|
# EXTERNAL_ID:                d4176972-bb0e-44b3-90d2-bb8e96222222
# ID:                          d4176972-bb0e-44b3-90d2-bb8e96222222
# IN_ANY_CONTEXT:              true
# NODE_TYPE:                   ELEMENT
# PROP_element_type_id:        COMPONENT
# PROP_meta_type:              WinSQLDatabase
# PROP_name:                   DB2From
# PROP_production:             1000
UNRECONCILED    d4176972-bb0e-44b3-90d2-bb8e96222222

# CUSTOM_STATE_PROVIDER:      {}
# DERIVED_STATE_AVAILABILITY:  __DERIVED_STATE__|UP|
# DERIVED_STATE_PERFORMANCE:  __DERIVED_STATE__|ACCEPTABLE|
# EXTERNAL_ID:                d66416a5-7db7-47df-b9ab-ed9422666666
# ID:                          d66416a5-7db7-47df-b9ab-ed9422666666
# IN_ANY_CONTEXT:              true
# NODE_TYPE:                   ELEMENT
# PROP_element_type_id:        COMPONENT
# PROP_meta_type:              WinSQLDatabase
# PROP_name:                   tempdbFrom
# PROP_production:             1000
UNRECONCILED    d66416a5-7db7-47df-b9ab-ed9422666666

# DERIVED_STATE_AVAILABILITY:  __DERIVED_STATE__|DOWN|
```

```
# DERIVED_STATE_PERFORMANCE:  __DERIVED_STATE__|ACCEPTABLE|
# DESCRIPTION:
# EXTERNAL_ID:                1f3e3f0b-35a4-4851-8863-b94f36AAAAAA
# ID:                          1f3e3f0b-35a4-4851-8863-b94f36AAAAAA
# IN_ANY_CONTEXT:              true
# NODE_TYPE:                   SERVICE
# ORGANIZER:                   /Zreconsvcs
# PROP_element_type_id:       SERVICE
# PROP_meta_type:              DynamicService
# PROP_name:                   DBSvcExpTarget
# RECONCILE_ACTION:           CREATE
CREATE      1f3e3f0b-35a4-4851-8863-b94f36AAAAAA

MAP      8ddcc69b-e6a7-4370-8f3d-4a53f897b9f4  /zport/dmd/Devices/Server/Microsoft/Windows/devices/10.111.5.81

MAP      fc6ab26d-75cd-491a-8c71-75fb25845f97  /zport/dmd/Devices/Server/Microsoft/Windows/devices/10.111.5.81/
os/winsqlinstances/INSTANCE1/databases/INSTANCE15
```

Adding an Impact Services portlet

You can customize the Resource Manager dashboard by adding an Impact Services portlet.

Service Impact must be installed and at least one dynamic service must exist.

1. In the Resource Manager browser interface, choose DASHBOARD.
2. In the top-right of the main dashboard area, click Add, and then choose Add portlet.
3. From the Portlet drop-down list, choose Impact Services.
4. In the Preview pane, choose the organizer or services that you want to show in the portlet, and then click Add.

If you delete the organizer displayed in a portlet, the portlet is deleted as well.

Production state propagation threshold

Service Impact relies on device production states to decide whether to propagate availability and performance states within a service graph.

The following table characterizes the device production states that are available in Resource Manager.

Production state	Devices monitored?	Appear on dashboard?
Production	Yes	Yes
Pre-Production	Yes	No
Test	Yes	No
Maintenance	Yes	Might appear
Decommissioned	No	No

Devices are displayed in the [Impact View](#) as determined by production state and production state propagation threshold. Each threshold setting indicates which states are considered to be "in production." For example:

- The threshold value Production indicates that only production-level devices are to be considered "in production."
- The threshold value Decommissioned indicates that devices in all production states are to be considered "in production."

In the Impact View, background color indicates the node's production state. In the following table,

- FALSE indicates that the node is displayed with a gray background, and its state is not propagated upwards to other members.
- TRUE indicates that the node is displayed with a white background, and interacts with other members as "in production."

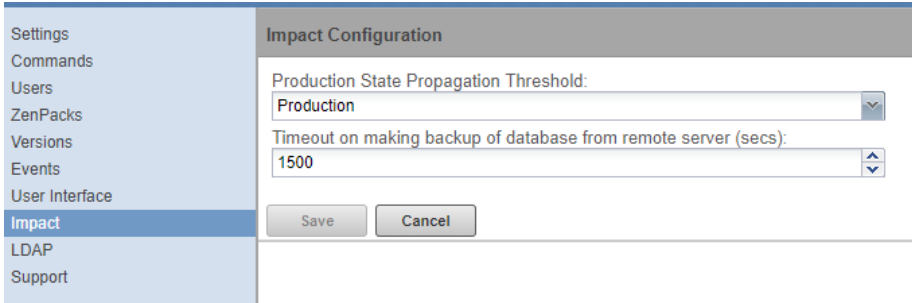
Device production state	Threshold= Production	Threshold= Pre-Production	Threshold= Test	Threshold= Maintenance	Threshold= Decommissioned
Decommissioned	FALSE	FALSE	FALSE	FALSE	TRUE
Maintenance	FALSE	FALSE	FALSE	TRUE	TRUE
Test	FALSE	FALSE	TRUE	TRUE	TRUE
Pre-Production	FALSE	TRUE	TRUE	TRUE	TRUE
Production	TRUE	TRUE	TRUE	TRUE	TRUE

For more information about production states, see [Production states and maintenance windows](#).

Configuring the production state propagation threshold

The default value of the Service Impact production state propagation threshold is Production. You can select a different production state for the threshold. Zenoss recommends changing this threshold only during installation.

1. Log in to the Resource Manager browser interface as a user with ZenManager or Manager privileges.
2. Select the ADVANCED tab, and then click the Impact link in the left column.



3. From the Production State Propagation Threshold list, select a new production state, and then click Save. Devices are evaluated against the new threshold at the next modelling interval. To update all devices before the next modelling interval, perform a graph update.

Service Impact server configuration files

The Service Impact server configuration files are stored in the Infrastructure/Impact container. Zenoss recommends that you edit configurations through the Control Center browser interface, which ensures that changes are preserved and pushed to new containers on restarts.

Use the Control Center browser interface to edit and preserve changes to `/opt/zenoss_impact/etc/logback.xml`. Modifying `logback.xml` in the Control Center browser interface requires a restart of the Service Impact service for the logging changes to take effect. If you use the command line to implement logging changes, they are implemented immediately; however, the changes made using the command line are lost when Service Impact is restarted.

To test changes in a pre-production environment, modify configuration file settings directly in the container. When Service Impact is restarted, all settings are returned to their default value.

Follow these steps to edit Service Impact configuration files:

1. Log in to the Control Center browser interface.
2. In the Applications table, click `Zenoss.resmgr`.
3. Scroll down to the Services list and select `Infrastructure/Impact`.
4. Under Configuration Files, click Edit to the right of `/opt/zenoss/etc/dsa.zenoss-dsa.properties`.
The `zenoss-dsa.properties` file is displayed in the Edit Configuration window.
Note: Modifying properties might change the performance of Service Impact. Before you modify default values, consult Zenoss Support.
5. Make changes and then click Save.
6. To activate the changes, restart Service Impact.

Removing Service Impact

Use this procedure to remove Service Impact from a Resource Manager 5.x or 6.x deployment.

1. Log in to the Control Center master host as a user with Control Center CLI privileges.
2. Create a snapshot:

```
serviced service snapshot Zenoss.resmgr
```

On completion, the `serviced` command returns the ID of the new snapshot. If this procedure fails, you can restore the snapshot created in this step. For more information about restoring a snapshot, see [Creating snapshots and rolling back](#).

3. Stop the Zenoss services, and then verify that the services are stopped.
 - a. Stop Resource Manager.

```
serviced service stop Zenoss.resmgr/Zenoss
```

- b. Wait until all services are stopped.
Use the `watch` command to monitor the status.

```
watch serviced service status Zenoss.resmgr/Zenoss
```

4. Start the services that are required for the removal of Service Impact.
 - a. Start the `zeneventserver` and `Zope` services.

```
serviced service start zeneventserver zope
```

- b. Wait until the services are started.
Use the `watch` command to monitor the status.

```
watch serviced service status zeneventserver zope
```

5. Remove the `ZenPacks.zenoss.Impact` ZenPack.
Removing the ZenPack also removes the `zenimpactstate` service.

```
serviced service run zope zenpack-manager uninstall ZenPacks.zenoss.Impact
```

6. Stop and remove the `Infrastructure/Impact` service, and then remove the `ZenPacks.zenoss.ImpactServer` ZenPack.
 - a. Stop the `Infrastructure/Impact` service, then wait 10 seconds.

```
serviced service stop Infrastructure/Impact
```

- b. Remove the `Infrastructure/Impact` service.

```
serviced service remove Infrastructure/Impact
```

- c. Remove the ZenPack.

```
serviced service run zope zenpack-manager uninstall ZenPacks.zenoss.ImpactServer
```

7. Restart all Zenoss services:
If you are updating to version 5.5.x, skip this step.

```
serviced service restart Zenoss.resmgr/Zenoss
```

Service Impact MIB file

Service Impact includes its own SNMP management information base (MIB) file. The file is located here:

```
/opt/zenoss/ZenPacks/ZenPacks.zenoss.Impact-VERSION-py2.7.egg/ZenPacks/zenoss/Impact/share/mibs/ZENOSS-IMPACT-MIB.txt
```

The file is a supplement to the standard Zenoss MIB file, containing definitions for events that are unique to Service Impact. To use this MIB file in other applications that receive traps, you must install both this file and the Zenoss MIB file.

Optionally, the events defined in this file may be used in [notifications](#).

Dynamic service name restrictions

Only the following characters are allowed in the names of dynamic services and dynamic service organizers:

- upper-case and lower-case letters of the Basic Latin character set (A-Z,a-z)
- digits (0-9)
- the space character ()
- the full stop character (.)
- the hyphen-minus character (-)
- the low line character (_)
- the number sign character (#)
- the dollar sign character (\$)
- the left parenthesis character (()
- the right parenthesis character ())

In object IDs, the number sign character (#) is replaced with the low line character (_). In titles, the number sign is unchanged.

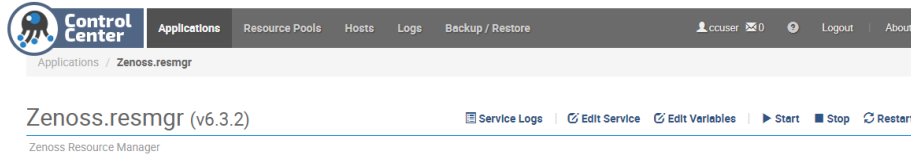
Configuring relationship processing

Use this procedure to configure Resource Manager and Service Impact to use lazy or eager relationship processing.

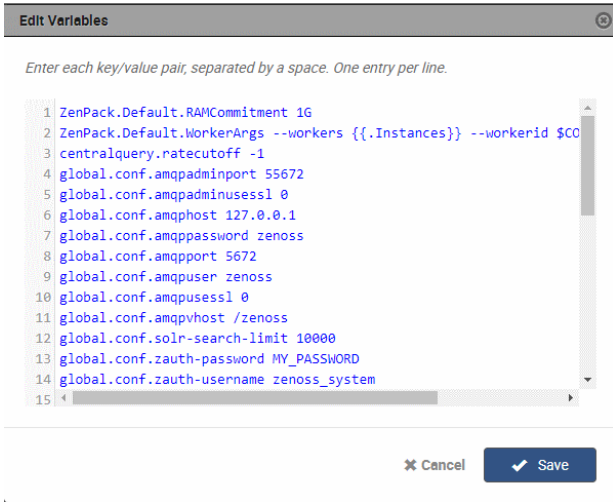
- Lazy mode is the default setting for release 5.5.x, and it is the appropriate choice when all of the ZenPacks that you use to monitor devices have been updated for optimal relationship processing.
- Eager mode ensures high-accuracy models at the potentially unacceptable cost of low remodeling performance. In large or complex environments, long remodeling times can generate inaccuracies in the model as changes to the underlying devices are processed. Eager mode is appropriate for customers with smaller and less complex models and a low rate of model changes.

Perform these steps:

1. Log in to the Control Center browser interface.
2. Navigate to the Applications > Zenoss.resmgr page.



3. On the application title line, click Edit Variables.



4. In the Edit Variables dialog box, add the following variable:

```
global.conf.corroborating-relationship-provider True
```

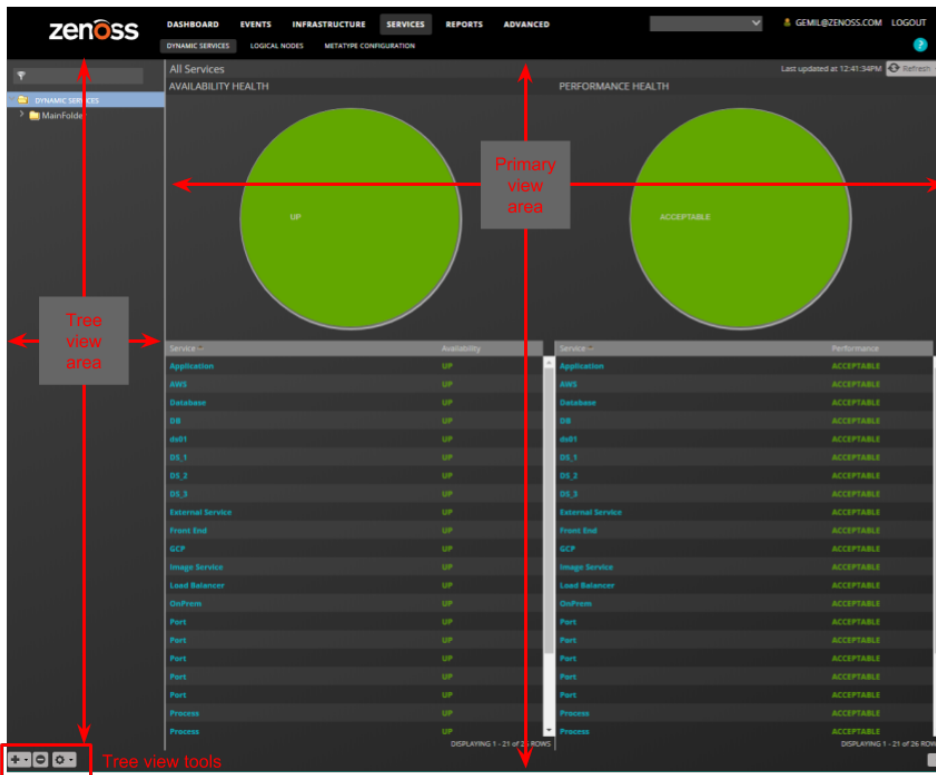
- To configure lazy relationship processing, set the value to `True`.
- To configure eager relationship processing, set the value to `False`.

If you switch the processing mode from lazy to eager (change the value from `True` to `False`), you must [rebuild the graph](#).

5. At the bottom of the Edit Variables dialog box, click Save.
6. On the application title line, click Restart.

Service Impact home page

The home page of the Service Impact feature is SERVICES > DYNAMIC SERVICES, which displays the health summaries of all dynamic services. From the home page, you can choose the LOGICAL NODES or METATYPE CONFIGURATION tabs.



Tree view area

The tree view area displays service nodes and logical nodes in alphabetical order. You can create organizers and order them as you wish. To move service nodes and logical nodes into organizers, drag them into the tree view.

Tool	Menu		
	DYNAMIC SERVICES	LOGICAL NODES	METATYPE CONFIGURATION
	Add Dynamic Service Add Dynamic Service Organizer	Add Logical Node Add Logical Node Organizer	(no menu)
	(no menu)	(no menu)	(no menu)
	View and Edit Details Clone Service... Export Selected	(no menu)	Restore default settings

Primary view area

The content of the primary view area depends on which tab is selected, and for some tabs, which item is selected in the tree view area.

Tab	Tree view selection	Primary view contents
DYNAMIC SERVICES	DYNAMIC SERVICES (the root organizer)	The availability and performance health summaries of all services.

	An organizer	The availability and performance health summaries of the services that the organizer contains.
	A dynamic service (service model)	The Members of the selected service. From this view, select Impact Events or Impact View .
LOGICAL NODES	<ul style="list-style-type: none"> • LOGICAL NODES (the root organizer) • An organizer 	A blank logical node details view .
	A logical node	The details view of the selected logical node. No other views are associated with logical nodes.
METATYPE CONFIGURATION	The list of ZenPacks with metatypes defined	The components that can be excluded from all dynamic services that include the components.

DYNAMIC SERVICES page

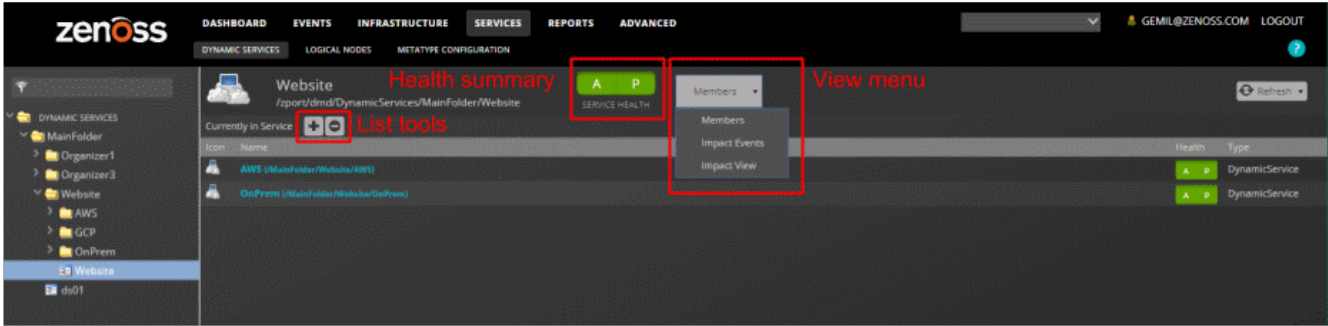
The SERVICES > DYNAMIC SERVICES page provides the most-frequently-used features of Service Impact.

- [Members view](#)
- [Add to Service dialog box](#)
- [Impact Events](#)
- [Impact View](#)
- [Impact Policies dialog box](#)

Members view



The Members view provides details about a service member, including the list of its associated members.

The following example shows the Members view of a service member, with key features highlighted, and the view menu selected.



List tools

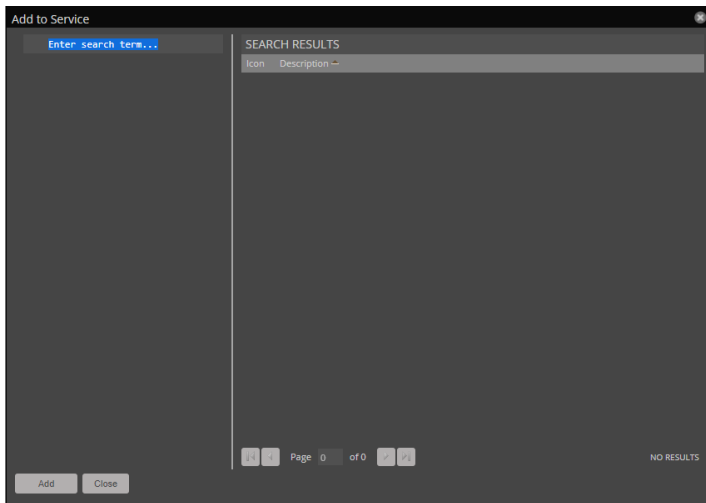
The list tools add or remove members from a service model.

Tool	Function
	Display the Add to Service dialog box.
	Remove a selected member from the service model.

Add to Service dialog box

Use the Add to Service dialog box to find and add members to a service model.

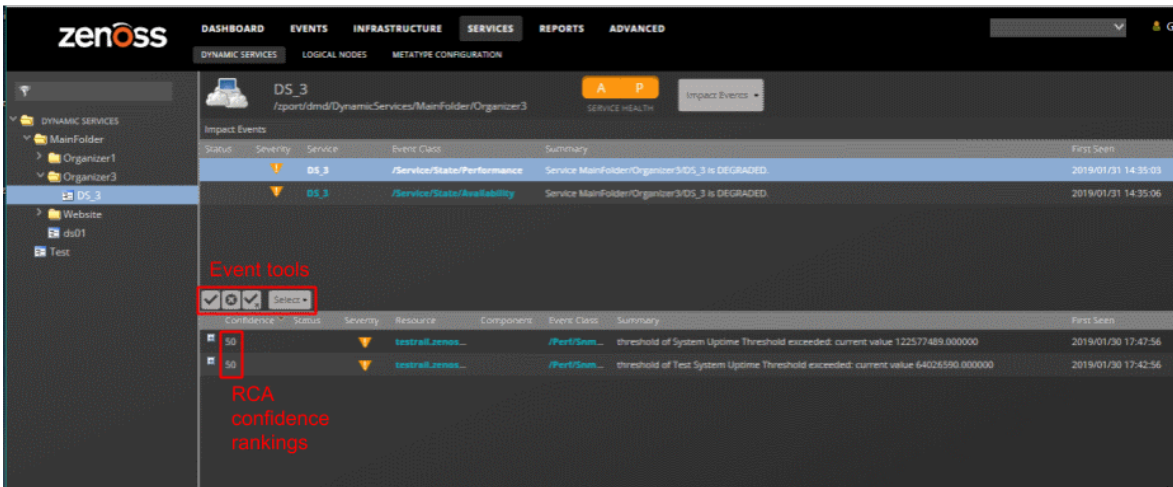
In the search field, enter the string to find. You can search for a specific device or component name, or component type (such as interfaces and OS processes) or group type (such as location and system). As you type characters, Service Impact displays matches.



Impact Events





The Impact Events view shows summary and detail information about events that are affecting a service.

The following example shows an Impact Events view with key features highlighted. The last event in the details area is expanded to show the Impact chain, which is the hierarchy of service model members that are associated with the event.



Event tools

The event tools provide options for manipulating the list of events.

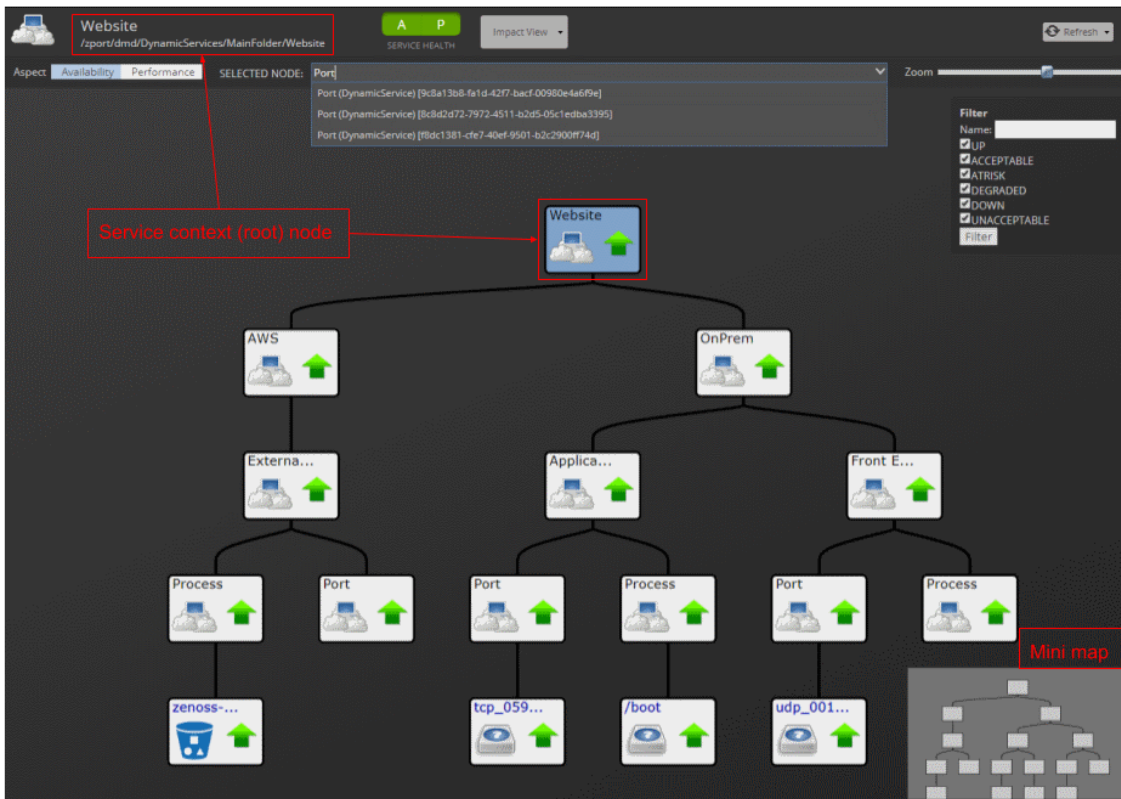
Tool	Function
	Acknowledge the selected event.
	Close the selected event. The event is moved to the archive.
	Undo acknowledgement of the selected event.
	Display the event selection menu. Select All Select all events in the list. None Deselect all selected events.

Root-cause analysis (RCA) confidence rankings

The second column of the details list contains the event's confidence ranking. Service Impact knows which members affect which service models, and automatically performs RCA when an event occurs. The analysis yields a probability value that an event is the cause of the service's current state. Often, multiple events contribute to a state, so the confidence rankings enable you to quickly focus resources on the most likely cause.

Impact View

The Impact View displays the interactive Service Impact graph of a service model and provides several views and tools.



Service context (root) node name

In the upper left above the Aspect controls, the service context (root) node of the graph is displayed, along with its tree view path. Click the name to make the root node the selected node and display its name in the Selected node field.

Aspect views: Availability, Performance

Use aspect views to display and filter on availability or performance states in the graph.

Selected Node

Displays the name and type of the node that you have chosen, or the service context (root) node if you have not chosen a node. For example, DB Hosts (Organizing Group) is the node named DB Hosts, which is an organizing group type. When you first access the Impact View, the Selected Node field displays the root node. To resize the Selected Node field, hover the pointer to the left of the field label to display the resize tool, and then drag the tool left or right. When you choose a node from the list or click a tile in the graph, that node appears in Selected Node and the node tile is centered in the graph.

To search for specific nodes, begin typing a string in the Selected Node field. The field's drop-down list displays matching nodes in the following format:

```
node_name (node_type) [UID]
```

Tips for searching nodes:

- Strings are not case sensitive.
- Regular expressions are not supported.
- You can search for nodes by name, type, or system-generated unique identifier (UID).
- To search for nodes by name only, suffix the search string with an open parenthesis. For example, the following string matches only node names that end with "Hosts," such as the DB Hosts node.

```
Hosts (
```

However, the following string matches nodes with "hosts" in the name or type.

hosts

- To search for nodes by type only, prefix the search string with an open parenthesis. For example, the following string matches only node types beginning with "vSphere," such as the vSpherePnic type.

(vSphere

However, the following string matches nodes with "vsphere" in the name or type.

vSphere

- To search for nodes that contain the same character string in the UID, prefix the search string with an open bracket. For example, the following string matches only nodes with a UID beginning with "G7M," such as G7M589.

[G7M

However, the following string matches nodes with "G7M" in the name, type, or UID.

G7M

Zoom

To increase or decrease the size of tiles in the graph, use the mouse wheel or drag the Zoom control in the upper right corner of the view.

Node tools

To see a summary of a node in a Service Impact graph, hover the pointer over its tile. Summary information includes the node name, availability state, performance status, production status, and type.

To display the following menu options for a node, position the pointer over it and right-click:

Toggle Children

Hide or display the node's child and descendent nodes.

Edit Impact Policies

Display the Impact Policies dialog box. See [Impact Policies dialog box](#). Note, Impact Policies for Availability and Performance are set separately, depending on which aspect is selected.

Center view on node

Adjust the view so the selected node is in the center of the display area.

Graph and filter tools (right-click menu)

To display the following Impact View menu options, position the pointer in the primary view area and right-click.

Fit Graph to Window

Adjust the size of the graph so that its dimensions match the display area.

Show All

Display all nodes in the graph, including those that are automatically added by ZenPacks.

Collapse All

Display only the top-level node of the graph.

Compact View

Display only the service model members that you have manually added and the immediate children of a service or service group.

Toggle Rainbows

Display colored tabs at the right edge of the node, indicating the number of critical, error, and warning events associated with the node.

Show Filters

Hide or display the filter controls.

Use filter check boxes to select availability and performance states by which to filter. By default, all states are selected for filtering. To restrict displayed nodes to specific states, such as DOWN and UNACCEPTABLE, select only those check boxes. (Changes to the selected states apply to all dynamic services because state is a global feature.)

To filter by complete or partial node name, use the Name field. To perform the filter click Filter.

Export Graph Image

Creates a file of the graph image with the default name graph.png and downloads it to your workstation.

Enable center view

Shift the graph so that the selected node is in the center. The zoom level does not change. You can zoom in and out on the node in the center of the graph. Choose this option to recenter a specific node.

Jump to Selected



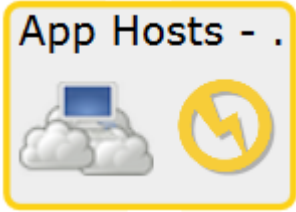
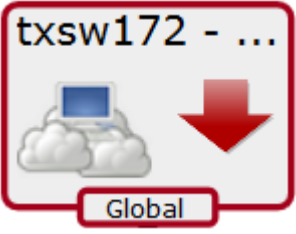
Shift the graph so that the node that is shown in the Selected Node field is in the center.

Jump to Context (root)

Shift the graph so that the service context (root) node of the graph is in the center.

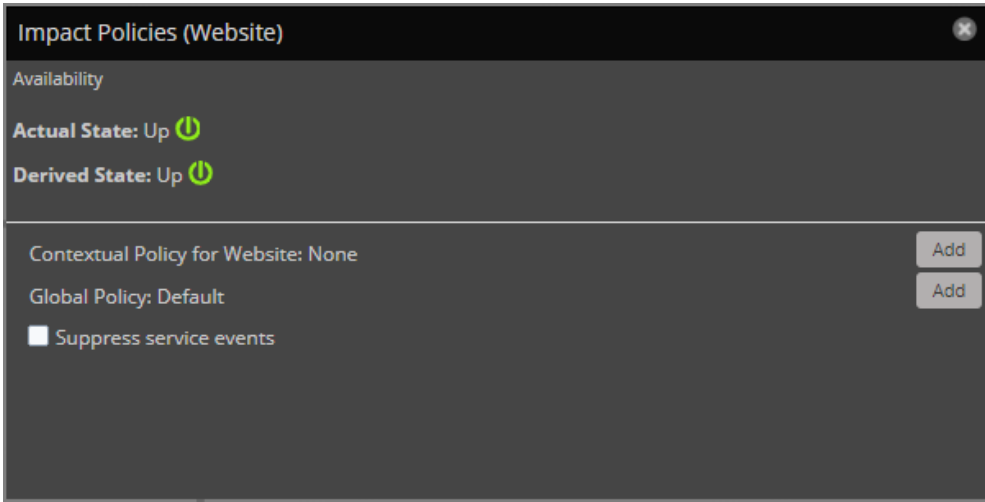
Node tile

The node tile graphically represents a server or service node and information about it, as described in the following table:

Node tile example	Description
	<p>This tile represents a Linux server named epowell-debug.zenoss.loc. (The name is shortened to fit inside the tile.)</p> <ul style="list-style-type: none"> • The computer icon and penguin image represent the node's type, Linux server. • The green arrow and the black border represent the node's availability state, UP.
	<p>This tile represents a service node with a global policy. Event rainbows are displayed.</p> <ul style="list-style-type: none"> • The event rainbow is the colored tabs on the right side of the tile. From top to bottom, the rainbow shows the counts of critical, error, and warning events that are associated with the node. The first example node has no events, so no numbers are displayed. The second example node has 1 critical error. • When you click an event rainbow tab, Collection Zone displays the overview page of the node. • This tile includes three dots immediately below its bottom border. The dots represent descendant nodes. To display the descendant nodes, double-click the node tile.
	<p>This tile represents a service node. The yellow icon and border represent the node's availability state, ATRISK.</p>
	<p>This tile represents a service node with a global policy. The red arrow and the red border of the tile represent the node's availability state, DOWN.</p>

Impact Policies dialog box

In the Impact Policies dialog box, add or edit state triggers for contextual or global policies. Note, Impact Policies for Availability and Performance are set separately, depending on which aspect is selected.



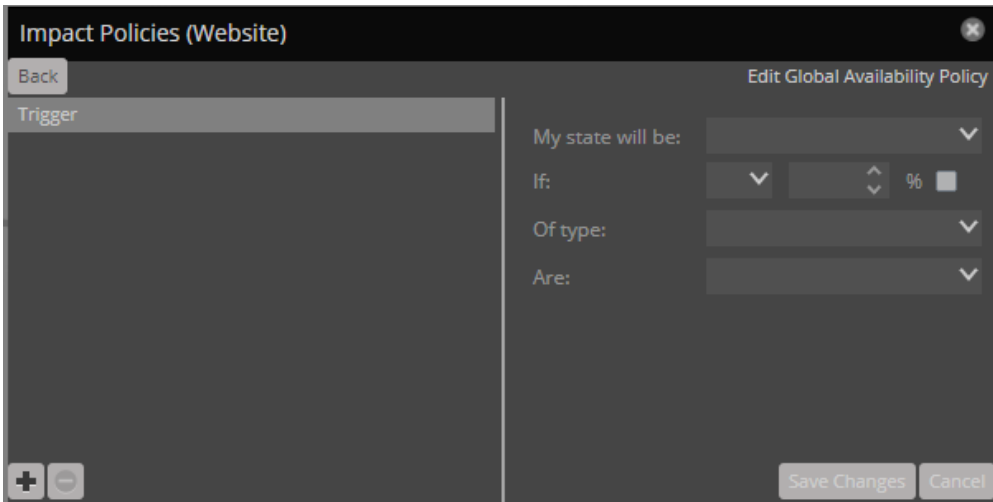
To prevent sending service events when changes affect a node, check Suppress service events. Choose this option when a service node is used solely to group child nodes.

If a custom state provider is not associated with a node, the dialog box does not contain the option to edit the custom state provider.

For a policy or custom state provider, click Add or Edit to access options.

Edit Policy options (Availability)

Use the Edit Policy options to add or edit state triggers for contextual or global policies.

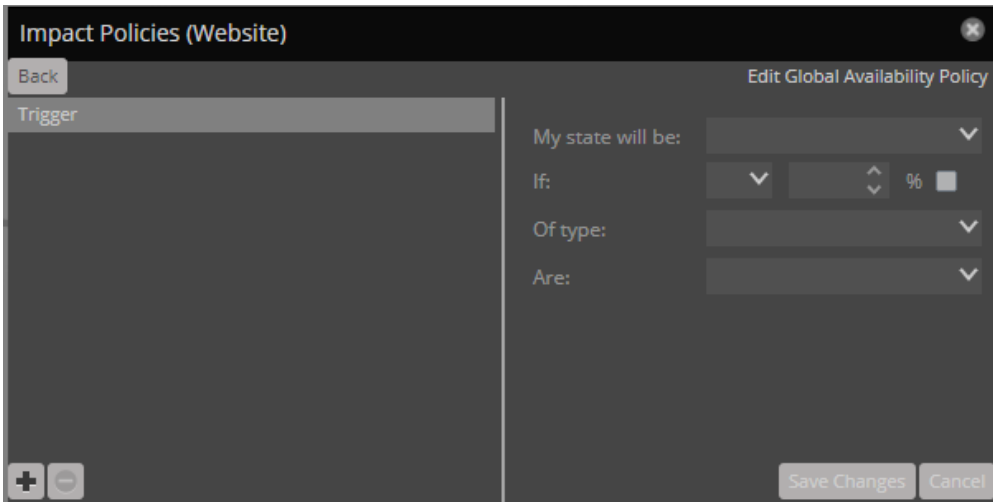


To add a trigger, in the lower-left corner click Add. On the right side of the dialog box, modify the trigger fields (described in the following table), and then click Save Changes.

Field	Description
My state will be	The new state, if the trigger applies. Possible states: DOWN, DEGRADED, ATRISK, UP.
If	The conditions that trigger a state change. To specify a percentage rather than an absolute value, click %.
Of type	Restrict the type of child node to which the trigger applies. Types other than Any are exclusive.
Are	The state of child nodes that cause an evaluation of this trigger. Possible states: DOWN, DEGRADED, ATRISK, UP.

Edit Policy options (Performance)

Use the Edit Policy options to add or edit state triggers for contextual or global policies.



To add a trigger, in the lower-left corner click Add. On the right side of the dialog box, modify the trigger fields (described in the following table), and then click Save Changes.

Field	Description
My state will be	The new state, if the trigger applies. Possible states: UNACCEPTABLE, DEGRADED, ACCEPTABLE.
If	The conditions that trigger a state change. To specify a percentage rather than an absolute value, click %.
Of type	Restrict the type of child node to which the trigger applies. Types other than Any are exclusive.
Are	The state of child nodes that cause an evaluation of this trigger. Possible states: UNACCEPTABLE, DEGRADED, ACCEPTABLE.

Edit Custom State Provider options

Use a custom state provider to add state triggers (rules) for custom device and component service model members. You can customize Resource Manager to gather state data from events that belong to other classes. For example, you can customize state providers to define state triggers for members that are monitored through customized classes that the ZenVMware and CiscoUCS ZenPacks provide.

Field	Description
Event Class	Choose the Resource Manager event class to monitor. You can configure one event class per device.
Event severity fields (Critical, Error, Warning, Info, Debug, Clear)	Choose the state for this member if the event severity is observed. Possible states are UP, ATRISK, DEGRADED, DOWN.
Apply to	<p>Choose the nodes (components) to which the state override applies.</p> <ul style="list-style-type: none"> • This node only: The selected component on the specific device. • Nodes of the same type on the same device: The same component type on the same device. If a component is associated with another device, it is not affected in that context. • Nodes of the same type in the same device class: The same component type on any device with the same device class. • Nodes of the same type system-wide: The same component type, regardless of the device or the device class.

LOGICAL NODES page

The SERVICES > LOGICAL NODES page allows you to create and edit customizable members that capture specific Collection Zone event states.

Logical nodes allow you to represent multiple resources or services; they rarely relate to events from a single member. For resources that Collection Zone monitors, logical nodes enable you to capture event states from arbitrary classes. Logical nodes might be external events from third-party monitoring products that are not monitored by Collection Zone.

For example, a bookstore website uses a third-party financial service to process credit cards. Collection Zone has no access to that third-party environment, but the monitoring systems forward events to Collection Zone. Their member names are prefixed with FincCC (that is, FincCC-AppSvr01, FincCC-AppSvr02, FincCC-OraDB01, and so on). To match events from that system and relate them to the "bookstore" service, you create a logical node that matches any event with a resource name that includes the prefix FincCC.

A logical node can be a child of a service model member, but no other type.

NAME:

DESCRIPTION:

PRODUCTION STATE:

CRITERIA:

all OF THE FOLLOWING RULES:

Resource name contains

Availability State:

Events for this node in this event class:

will result in these availability states:

Critical: Error: Warning: Info: Debug: Clear:

Performance State:

Events for this node in this event class:

will result in these performance states:

Critical: Error: Warning: Info: Debug: Clear:

Save Cancel

Criteria

The Criteria rules allow you to define event triggers and associate the triggers with availability and performance states.

Using rules that include numeric comparisons—in particular, greater than and less than comparisons—requires a clear understanding of the values that are included in source events. For example, severity values sent by [syslog use a scale of 0 to 7](#), with 7 being the least severe. This is the reverse of the scale that Collection Zone events use.

Availability state

In the "Events for this node in this event class" field, specify the event class or subclass that is associated with the trigger.

The following examples illustrate how to specify event classes.

/Status

Only the /Status class.

/Status/Web

Only the Web subclass of /Status

/Status/

The /Status class and all of its subclasses.

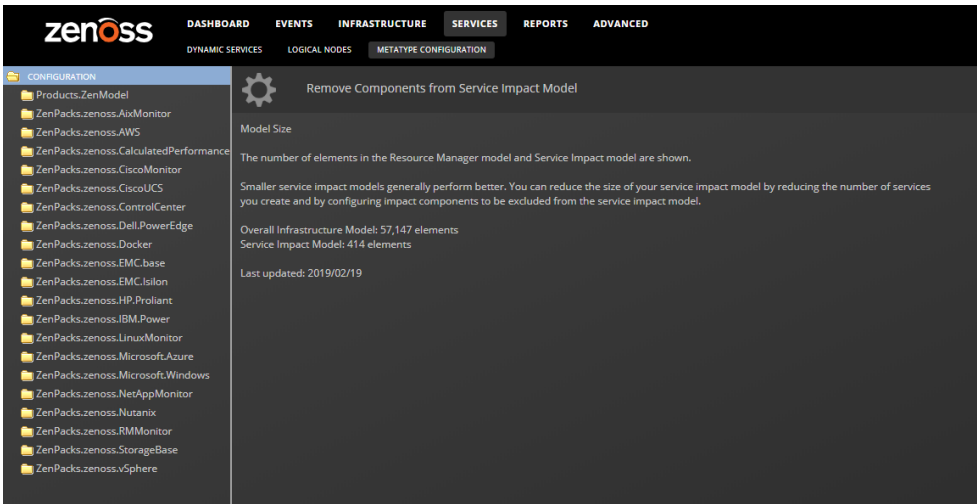
In the "will result in these availability states" field, map event severity levels to availability states.

Performance State

The fields and options in this area differ only in that the mapping of event severity levels to performance states uses different states.

METATYPE CONFIGURATION page

The configuration options on this page remove components from all dynamic service models. Every ZenPack that includes Service Impact relationships is displayed here.



If your Service Impact model is relatively large, you can improve the speed of both Service Impact and Resource Manager by removing unimportant components. For example, you may choose to exclude the server fan or power supply components from virtual machines. When a component is removed in this way, it no longer appears in the Impact View of dynamic services, and component events are not considered in determining the performance or availability of dynamic services.

Zenoss strongly recommends that you contact Zenoss Support to determine whether your environment can benefit by removing components, and to select components to remove.

Release notes

Release dates

Release	Date
5.5.0 CA	Jun 2019
5.4.8	Mar 2019
5.4.7 CA	Mar 2019
5.4.6	Mar 2019
5.4.5 CA	Feb 2019
5.3.1	Apr 2018

Resource Manager compatibility

Service Impact	Resource Manager
5.5.0 CA	6.3.2
5.4.8	
5.4.7 CA	
5.4.6	
5.4.5 CA	
5.3.1	6.x
	5.2.x and 5.3.x

ZenPack compatibility

Release	ZenPacks.zenoss.ImpactServer	ZenPacks.zenoss.Impact
5.5.0 CA	5.5.0.0.0	5.5.0.0.0
5.4.8	5.4.8.0.0	5.4.8.0.0
5.4.7 CA	5.4.7.0.0	5.4.7.0.0
5.4.6	5.4.6.0.0	5.4.6.0.0
5.4.5 CA	5.4.5.0.0	5.4.5.0.1
5.3.1	5.3.1.0.0	5.3.1.0.0

Detailed release notes

- [Service Impact 5.5.1](#)
- [Service Impact 5.5.0 CA](#)
- [Service Impact 5.4.8](#)
- [Service Impact 5.4.7 CA](#)
- [Service Impact 5.4.6](#)
- [Service Impact 5.4.5 CA](#)
- [Service Impact 5.3.1](#)

Service Impact 5.5.0 CA

This is a controlled-availability release. If you are interested in updating to this release, please contact your Zenoss representative.

This release improves overall performance, better handles commit conflicts, simplifies relationship building, and reduces the time needed to change production states. To support this release, many ZenPacks were updated, to improve performance in defining Service Impact relationships, and reduce inconsistencies by consolidating edge providers with the DynamicView ZenPack.

Update instructions

This release supports the following, separate update paths:

- Update from 5.1.x, 5.2.x, or 5.3.x to 5.5.0
- Update from 5.4.x to 5.5.0

Both update paths start at [the same page](#).

The order of steps in the update procedures is crucial. Please review the procedures in advance, and follow the instructions carefully.

Known issues

- In previous releases, adding a dynamic service was very quick, because the Service Impact database included all of the devices and components in the ZODB. In this release, Service Impact creates a background job to copy the devices and components it requires, which can take a few minutes to complete. Scripts that utilize the Service Impact JSON API must be updated to wait for the job to complete before proceeding.
- When you change the metatype configuration, the Service Impact database is updated to add or remove nodes. Depending on which items are changed, processing the changes can take 30-90 minutes in very large environments.
- (IMP-740) During a graph update, the `zenimpactgraph` script creates one worker process for each available CPU core. Each worker requires up to 2GB of main memory. Without it, the host where the update is running may become unstable.
- (ZEN-31884) Occasionally, graph update fails with `ModelCatalogError: Exception performing search`. The workaround is to re-run the update.
- After installing one or more ZenPacks that have updated relationship providers, rebuild the graph. For more information, see [Rebuilding the graph database](#).

Fixed issues

ID	Description
IMP-124	Exporting an empty dynamic service organizer results in a traceback.
IMP-405	When debug logging is enabled, the log files consume all available space within 10 minutes and crash the Impact container.
IMP-412	Impact diagrams show the wrong content in the icon type field.
IMP-425	State change events stop processing, and a null pointer exception is thrown.
IMP-430	Performance during graph update deteriorates under certain conditions.
IMP-452	Graph updates cannot handle a large number of interlocking changes at the same time.
IMP-454	The upgrade script does not properly handle the starting version information.
IMP-688	Solr timeouts during graph updates.
IMP-699	The upgrade to 5.4.x silently removes leading spaces from dynamic service names, which changes their sort order.
IMP-711	The relationship edges built during graph update are not cached, which slows overall update performance.
IMP-716	Building relationships is slower in 5.4.x than in 5.3.x.
IMP-720	Impact state is calculated using event tags, not elements or subelements.
IMP-724	Adding a device organizer generates asymmetric relationships.
IMP-727	Model changes can cause lengthy graph updates.
IMP-731	Relationships among Cisco UCS devices are missing (Impact 5.4.x installed).
IMP-734	Model change events that do not affect any dynamic services are being sent to Impact for processing.
IMP-740	Graph update consumes more memory until memory is exhausted and the container crashes.

Service Impact 5.4.8

This release includes a major new feature, selective synchronization.

In releases before 5.4.1, the Resource Manager Zope Object Database (ZODB) and the Service Impact graph database were tightly synchronized, and all of the relationships among Resource Manager devices and components in your environment (the "model") were copied to the Service Impact graph database. In larger environments, when Resource Manager model updates were underway, the synchronization process reduced performance in Resource Manager and caused Service Impact to make analyses with incomplete data.

In this release, you can refine the list of components that are copied from ZODB to Service Impact. This approach works well in environments where the critical services defined in Service Impact are a relatively small part of the Resource Manager model.

Additional highlights of this release:

- The SERVICES > Metatype Configuration page is the new interface for deciding which device components to exclude from Service Impact. The page includes counts of the overall number of entities in the Resource Manager model and the Service Impact model, updated regularly.
- The export/import/reconciliation process has been re-implemented to improve accuracy, reduce reconciliation conflicts, and provide progress information.
- The `zenimpactgraph run --update` process has been re-implemented to scale out to the available host resources. The update time is reduced from days to hours.
- The `zenimpactworker` process is no longer necessary.

Update instructions

Updating from releases before 5.4.x

The update procedures for this release are unique and a special section has been created for them:

[Updating Service Impact](#)

The order of steps in the update procedures is crucial. Please review the procedures in advance, and follow the instructions carefully.

Updating from 5.4.x

To update Service Impact to this release from a previous release of 5.4.x, follow the standard update procedure:

[Updating Service Impact](#)

When the update is complete, run the following command to synchronize `impact-server`:

```
zenimpactgraph run -x push
```

Known issues

- In previous releases, adding a dynamic service was very quick, because the Service Impact database included all of the devices and components in the ZODB. In this release, Service Impact creates a background job to copy the devices and components it requires, which can take a few minutes to complete. Scripts that utilize the Service Impact JSON API must be updated to wait for the job to complete before proceeding.
- When you change the metatype configuration, the Service Impact database is updated to add or remove nodes. Depending on which items are changed, processing the changes can take 30-90 minutes in very large environments.
- (IMP-699) The upgrade to 5.4.x silently removes leading spaces from dynamic service names, which changes their sort order. To work around the issue, rename the services after the upgrade.
- (IMP-709) When logged in as Admin, changing the Impact View from performance to availability briefly raises a server exception dialog.

Fixed issues

ID	Description
IMP-649	Dynamic service names and organizer names must be updated manually to meet new requirements
IMP-676	Sequential API calls to <code>addToDynamicService</code> results in <code>zenjobs</code> hanging
IMP-700	The Service Impact view includes more virtual machines than are defined in a dynamic service
IMP-707	<code>zenimpactgraph</code> throws a <code>Traceback</code> , global name <code>'IRelationshipNode'</code> is not defined

Service Impact 5.4.7 CA

This is a controlled-availability release. If you are interested in updating to this release, please contact your Zenoss representative.

This release includes a major new feature, selective synchronization.

In releases before 5.4.1, the Resource Manager Zope Object Database (ZODB) and the Service Impact graph database were tightly synchronized, and all of the relationships among Resource Manager devices and components in your environment (the "model") were copied to the Service Impact graph database. In larger environments, when Resource Manager model updates were underway, the synchronization process reduced performance in Resource Manager and caused Service Impact to make analyses with incomplete data.

In this release, you can refine the list of components that are copied from ZODB to Service Impact. This approach works well in environments where the critical services defined in Service Impact are a relatively small part of the Resource Manager model.

Additional highlights of this release:

- The SERVICES > Metatype Configuration page is the new interface for deciding which device components to exclude from Service Impact. The page includes counts of the overall number of entities in the Resource Manager model and the Service Impact model, updated regularly.
- The export/import/reconciliation process has been re-implemented to improve accuracy, reduce reconciliation conflicts, and provide progress information.
- The `zenimpactgraph run --update` process has been re-implemented to scale out to the available host resources. The update time is reduced from days to hours.
- The `zenimpactworker` process is no longer necessary.

Update instructions

Updating from releases before 5.4.x

The update procedures for this release are unique and a special section has been created for them:

[Updating Service Impact](#)

The order of steps in the update procedures is crucial. Please review the procedures in advance, and follow the instructions carefully.

Updating from 5.4.x

To update Service Impact to this release from a previous release of 5.4.x, follow the standard update procedure:

[Updating Service Impact](#)

When the update is complete, run the following command to synchronize `impact-server`:

```
zenimpactgraph run -x push
```

Known issues

- In previous releases, adding a dynamic service was very quick, because the Service Impact database included all of the devices and components in the ZODB. In this release, Service Impact creates a background job to copy the devices and components it requires, which can take a few minutes to complete. Scripts that utilize the Service Impact JSON API must be updated to wait for the job to complete before proceeding.
- When you change the metatype configuration, the Service Impact database is updated to add or remove nodes. Depending on which items are changed, processing the changes can take 30-90 minutes in very large environments.
- (IMP-699) The upgrade to 5.4.x silently removes leading spaces from dynamic service names, which changes their sort order. To work around the issue, rename the services after the upgrade.

Fixed issues

ID	Description
IMP-690	Unable to export graph image when using Chrome
IMP-693, IMP-694	The state of a dynamic service is inconsistent when the production state of a member device is changed in Resource Manager

Service Impact 5.4.6

This release includes a major new feature, selective synchronization.

In releases before 5.4.1, the Resource Manager Zope Object Database (ZODB) and the Service Impact graph database were tightly synchronized, and all of the relationships among Resource Manager devices and components in your environment (the "model") were copied to the Service Impact graph database. In larger environments, when Resource Manager model updates were underway, the synchronization process reduced performance in Resource Manager and caused Service Impact to make analyses with incomplete data.

In this release, you can refine the list of components that are copied from ZODB to Service Impact. This approach works well in environments where the critical services defined in Service Impact are a relatively small part of the Resource Manager model.

Additional highlights of this release:

- The SERVICES > Metatype Configuration page is the new interface for deciding which device components to exclude from Service Impact. The page includes counts of the overall number of entities in the Resource Manager model and the Service Impact model, updated regularly.
- The export/import/reconciliation process has been re-implemented to improve accuracy, reduce reconciliation conflicts, and provide progress information.
- The `zenimpactgraph run --update` process has been re-implemented to scale out to the available host resources. The update time is reduced from days to hours.
- The `zenimpactworker` process is no longer necessary.

Update instructions

The update procedures for this release are unique and a special section has been created for them:

[Updating Service Impact](#)

The order of steps in the update procedures is crucial. Please review the procedures in advance, and follow the instructions carefully.

Known issues

- In previous releases, adding a dynamic service was very quick, because the Service Impact database included all of the devices and components in the ZODB. In this release, Service Impact creates a background job to copy the devices and components it requires, which can take a few minutes to complete. Scripts that utilize the Service Impact JSON API must be updated to wait for the job to complete before proceeding.
- When you change the metatype configuration, the Service Impact database is updated to add or remove nodes. Depending on which items are changed, processing the changes can take 30-90 minutes in very large environments.
- (IMP-676) Sequential calls in the Zenoss Impact API do not complete successfully when the number of jobs that can be processed concurrently is greater than one. To work around the issue, set the `concurrent-jobs` directive in the `/opt/zenoss/etc/zenjobs.conf` file to 1.
- (IMP-699) The upgrade to 5.4.x silently removes leading spaces from dynamic service names, which changes their sort order. To work around the issue, rename the services after the upgrade.

Fixed issues

ID	Description
IMP-641	Unexpected behavior from the discard changes popup
IMP-664	Cloning a dynamic service does not work
IMP-686, IMP-689	Nodes with policies are not properly imported
IMP-687	Unable to view or edit policies
IMP-680	Error flare in Impact View, Attribute Error "NoneType" object has no attribute 'dmd'

Service Impact 5.4.5 CA

This is a controlled-availability release. If you are interested in updating to this release, please contact your Zenoss representative.

This release introduces a major new feature, selective synchronization.

In previous releases, the Resource Manager Zope Object Database (ZODB) and the Service Impact graph database were tightly synchronized, and all of the relationships among Resource Manager devices and components in your environment (the "model") were copied to the Service Impact graph database. In larger environments, when Resource Manager model updates were underway, the synchronization process reduced performance in Resource Manager and caused Service Impact to make analyses with incomplete data.

In this release, you can refine the list of components that are copied from ZODB to Service Impact. This approach works well in environments where the critical services defined in Service Impact are a relatively small part of the Resource Manager model.

Additional highlights of this release:

- The SERVICES > Metatype Configuration page is the new interface for deciding which device components to exclude from Service Impact. The page includes counts of the overall number of entities in the Resource Manager model and the Service Impact model, updated regularly.
- The export/import/reconciliation process has been re-implemented to improve accuracy, reduce reconciliation conflicts, and provide progress information.
- The zenimpactworker process is no longer necessary.

Update instructions

The update procedures for this release are unique and a special section has been created for them:

[Updating Service Impact](#)

The order of steps in the update procedures is crucial. Please review the procedures in advance, and follow the instructions carefully.

Known issues

- In previous releases, adding a dynamic service was very quick, because the Service Impact database included all of the devices and components in the ZODB. In this release, Service Impact creates a background job to copy the devices and components it requires, which can take a few minutes to complete. Scripts that utilize the Service Impact JSON API must be updated to wait for the job to complete before proceeding.
- When you change the metatype configuration, the Service Impact database is updated to add or remove nodes. Depending on which items are changed, processing the changes can take 30-90 minutes in very large environments.
- (IMP-306) The list of allowed characters in the names of dynamic services and dynamic service organizers is constrained. For more information, see [Preparing to update to 5.5.x](#).
- (IMP-685) During the `zenimpactgraph run --update` process, you may receive a message similar to the following:

```
WARNING zen.ImpactGraph: Exception while adding impact batch:'NoneType' object has no attribute
'__module__', number tried: 9 (a retry number > 5)
```

Repeat the `zenimpactgraph run -x catclean` and retry the `zenimpactgraph run --update` process again.

Fixed issues

ID	Description
IMP-69	The criteria for logical nodes must consider the numbering system of source events
IMP-276	The health check for webservice-status fails, which causes additional failures
IMP-306	Special characters not supported in service or organizer names
IMP-399	(IMP-557) Reconciliation does not consider relationships in imported graph
IMP-425	Null pointer exception in Neo4j before state change events stop processing
IMP-431	The zenimpactworker service times out during long graph updates
IMP-432	zenimpactgraph loses connection to zodb after a long graph update
IMP-435	Both device and component production state are sent to the impact server
IMP-438	Exports do not include logical nodes that are not part of a dynamic service
IMP-444	Inconsistent production states when graph updates are run during a maintenance window

IMP-450	No retry when certain add node operations fail
IMP-451	Rollback exception not handled properly
IMP-452	updateGraph API cannot support large number of interlocking changes in one transaction
IMP-459	Upgrading the LinuxMonitor ZenPack fails due to large number of model changes at once
IMP-465	Strange behavior when following dynamic service member link
IMP-475	Can't add a service or organizer because Submit button is inactive
IMP-476	The Impact view does not support the dark theme in the Resource Manager browser interface
IMP-482	In some cases, Impact events are not shown on a service's Events view
IMP-490	The Impact View does not immediately show updates to the model
IMP-514	A warning message is displayed constantly during normal use
IMP-543	The upgrade.sh script is not working properly
IMP-552	Impact does not provide accurate data to the Zenoss call home feature
IMP-559	Loading search results is slow on larger devices with many components
IMP-572	Value for check-box in Suppress service events is not persistent
IMP-573	Changing a service name does not propagate everywhere
IMP-585	Error when attempting to save a policy with empty fields
IMP-586	Policies are not included in service export/import
IMP-588	Import process does not include progress information
IMP-589	Production state changes during maintenance windows are not reflected in Impact
IMP-603	Events that match logical node criteria aren't processed
IMP-616	Font color in Impact portlet makes text difficult to read
IMP-625	Removing Impact components takes many seconds with no UI notification
IMP-651	Indexing is slow
IMP-654	Unable to click an object in the Impact View and go to the device in Resource Manager
IMP-648, IMP-657	Graph update takes a long time on a large system
IMP-666	Warning message <code>Could not adapt</code> during graph update
IMP-671	ZenPack update fails due to Impact state changes
IMP-672	Restoring omitted components is not working
IMP-673	Unable to create file for edges due to missing parent directory

Service Impact 5.3.1

This release features the enhancements that were introduced in version 5.3.0:

- Faster graph updates and state propagation in environments with many nodes.
- A new right-click menu option enables you to center the Impact View on a node that you select. The graph shifts to display the selected node in the center, but does not change the zoom level. The new Selected Node field enables you to select from a list of nodes that match the partial text string that you specify. Enhanced filtering enables you to find and view nodes by availability and performance states and partial name string.

The following table identifies the files to download for each supported installation or upgrade scenario.

Resource Manager	Required files
6.x	ZenPacks.zenoss.ImpactServer-5.3.1.0.0-py2.7.egg
5.3.x	ZenPacks.zenoss.Impact-5.3.1.0.0-py2.7.egg
5.2.x	install-zenoss-impact_5.3_5.3.1.0.0.run

Download files from delivery.zenoss.com. Use your [Zenoss Support](#) credentials to log in.

Fixed issues

ID	Description
IMP-39	GraphML file that was exported from an older version of Service Impact cannot be imported into latest version.
IMP-284	Implicit bug in setting event state during propagation process.
IMP-285	Improve performance and responsiveness under high workload environments.
IMP-290	Do batching of production state updates in Service Impact server.
IMP-307	In a large database, cannot remove nonexistent nodes because requests time out.
IMP-309	Impact state propagation takes too long on devices with thousands of components.
IMP-316	Multithreaded implementation of related event processing.
IMP-320	Unable to delete top level logical node organizer when it contains children.
IMP-321	A state node accumulates best state events.
IMP-322	Import of a GraphML file associates incorrect devices to components.
IMP-323	Need to repair Neo4j objects with large number of non-relevant events.
IMP-349	Refactor handling Neo4j resources in multi-threaded environment.
IMP-351	When updating a graph, errors occur for some ZenPack objects.
IMP-357	State propagation process takes an unacceptable amount of time on devices with thousands of components.
IMP-369	graphreset (update) does not prune orphaned nodes.

Considerations and workarounds

Reidentifying devices

Resource Manager provides the ability to reidentify devices. The reidentification process deletes and re-adds the device with a new ID and GUID. When the GUID is changed, the device is removed from that particular Service Impact service. If you reidentify a device, you must manually re-add the device to the service model.

Impact graph update best practices

Graph updates and maintenance window operations use the same target graph database. Concurrent access of shared data increases time required to complete the operations, risk of transaction rollbacks and deadlocks, and risk of inconsistent state updates to the graph database. For best performance when updating an impact graph,

- Do not manually rebuild an impact graph during a time that overlaps a maintenance window or another graph update operation.
- Do not import dynamic service models while a graph update operation is in progress.

For more information, see [Manually rebuilding an impact graph](#).

Upgrade considerations

When you initially upgrade from Service Impact 5.2.3 to 5.3.1, the new version of the product removes accumulated event history that is no longer needed. A temporary performance slowdown might occur while nodes are read.

Limitations

After Service Impact is installed, Service Impact and Resource Manager are interdependent. However, Resource Manager issues affect Service Impact more frequently than Service Impact issues affect Resource Manager. For this reason, the list of known issues for a given Service Impact release might include items that manifest in Service Impact but are not caused by Service Impact software. Such items are noted in the list of known issues.