

# Experimental generic redundancy object

## “Alternative”

Offered under GNU General Public License v3.0; please keep in mind I am not an experienced iTop programmer and this object may have some unanticipated side effects.

iTop CMDB out of the box recognizes a few types of redundancies:

- Power for physical devices
- Virtual hosts within a Farm
- CIs within an ApplicationSolution

These are numerous other situations where redundancies are desirable, examples being:

- For a UPS, the number of batteries that must be live OR an outside power connection that must be live
- For a physical device, multiple redundant network connections
- For an ISCSI SAN, the enclosures are frequently homed to multiple redundant controllers

Adding enough AttributeRedundancySettings fields to cover all the possible relations – and the corresponding relationship neighborhoods behind them – is a lot of work with no guarantee of anticipating all possible relationships that a user might want to create.

I looked at creating an AttributeRedundancySettings to mimic that used in the ApplicationSolution class for its required underlying CIs. This doesn't work very well because all of the CIs are treated as a group. For example, it's pretty common to have redundant network connections and pretty common to have redundant SAN connections with the requirement that at least one of each be up. In other words:

(NetworkConnection A OR NetworkConnection B)  
AND  
(SANConnection A OR SANConnection B) with both groups up

Since ApplicationSolution/CIs don't allow distinguishing between groups of things, the best you can do is

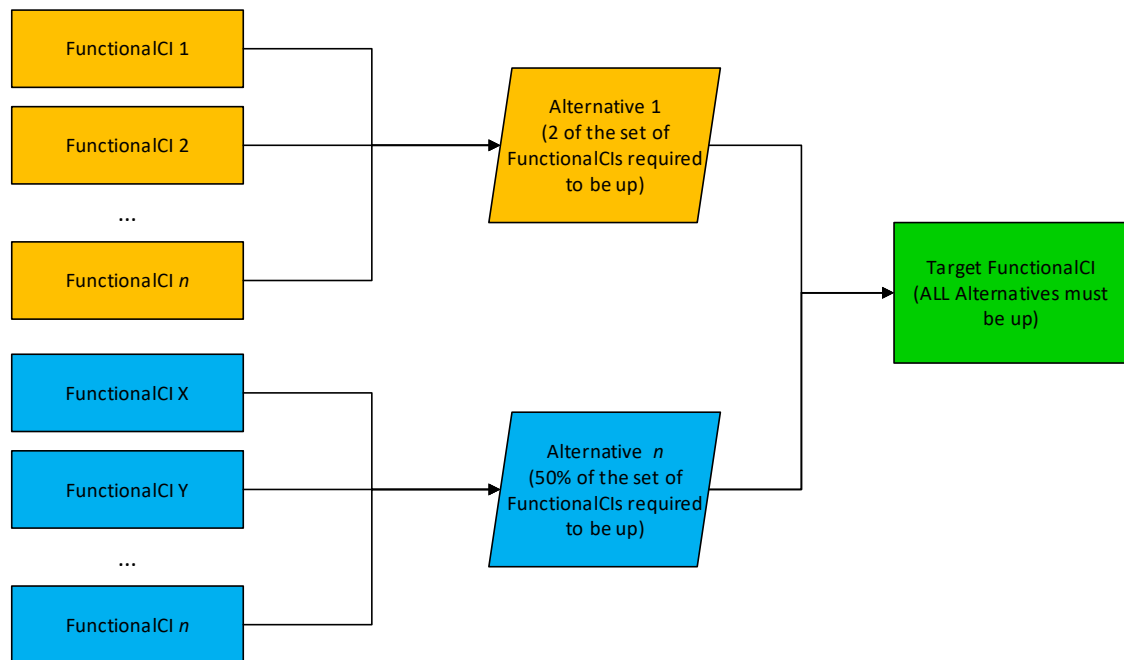
NetworkConnection A AND  
NetworkConnection B AND  
SANConnection A AND  
SANConnection B with all up

To allow for more complex AND/OR logic between objects, I built the Alternative object. The Alternative object binds together the objects that truly are alternates to one another - NetworkConnection A and NetworkConnection B for example – with a redundancy unique to that Alternative. The Alternatives are then bound to the dependent object as “all required”. So the example above becomes:

Alternative 1: (NetworkConnection A OR NetworkConnection B), 1 required

Alternative 2: ((SANConnection A OR SANConnection B), 1 required

Target FunctionalCI: (Alternative 1 AND Alternative 2), all required



Alternatives contain FunctionalCIs for objects that can be used as substitutes or alternates for one another.

The Alternative defines the redundancy requirement for the set of FunctionalCIs in the Alternative.

The Alternative can test any number of FunctionalCIs.

ALL the Alternatives have to be met in order for the target FunctionalCI to be up – Alternatives specify the redundancy or their substitutes.

The target can be dependent on any number of Alternatives

Generally, the instances bound together in an Alternative are the same class but it's not a requirement. For example, since BusinessProcess is also a FunctionalCI, you can build in a manual workaround by making the workaround a BusinessProcess and including it as one of the options of an Alternative.

At this point in development, Alternatives are not subclassed from FunctionalCI so you can't do an Alternative that contains other Alternatives.

## Installation

**DO THIS ON A TEST MACHINE. I AM NOT AN EXPERIENCED iTOP PROGRAMMER AND THIS IS AN EXPERIMENTAL CLASS.**

1. Copy ct-alternative to the extensions directory, run Setup and select it. I'm running version 2.6.0, I don't know if it will work on earlier versions.
2. For each FunctionalCI class where the Alternative class might use the FunctionalCI as a dependency or for any FunctionalCI class that might be the target of an Alternative, insert items in the <detail> presentation to create tabs for the used-in and required Alternatives:

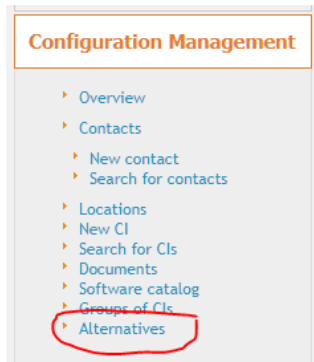
```
<item id="alternative_list">
  <rank>220</rank>
</item>
<item id="requiredalternatives_list">
  <rank>230</rank>
</item>
```

(Sorry this is a manual process – I am not yet confident enough in iTop programming to attempt to modify every subclass of FunctionalCI as part of this XML).

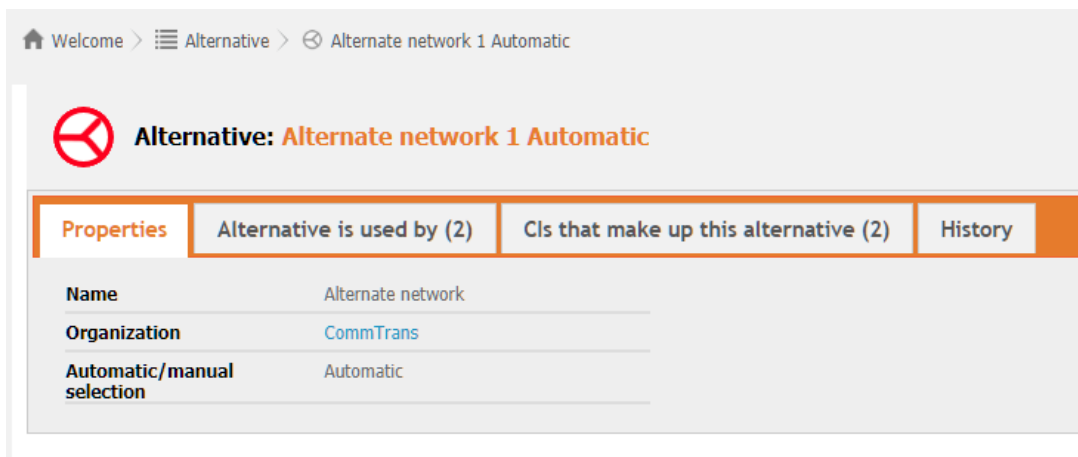
3. alternative\_list shows where a particular FunctionalCI instance is used by an Alternative. requiredalternatives\_list shows the list of Alternatives the current FunctionalCI instance depends upon.
4. Existing redundancies – power, ApplicationSolution CIs, hosts in a FARM – are not affected nor converted.

## Use

### 1. Creating an Alternative



2. Detail screen – the Alternative has a name and an owner. The display name is composed of 3 parts:
- The name
  - The redundancy. In the example, the 'disabled' in the name comes from the redundancy required all the objects be present. If you specify a count, this will be an integer, if you specify a percentage it will be a percentage.
  - The word "Automatic" or "Manual"; this indicates whether manual action is required to select the underlying FunctionalCIs (e.g., pull a plug and move it elsewhere) or whether the selection is automatic (for example, HA failover)




3. Alternatives are made up of FunctionalCIs; for the Alternative to be “up”, the redundancy condition for the underlying FunctionalCIs has to be met

**Modification of Alternative: Alternate network 1 Automatic**

Cancel Apply

Properties Alternative is used by (2) CIs that make up this alternative (2)



<input type="checkbox"/>	Functional CI	Organization
<input type="checkbox"/>	Network Device 395	CommTrans
<input type="checkbox"/>	Network Device 396	CommTrans

Remove selected objects Add Functional CI objects...

**Redundancy**

☐ The alternative is satisfied if all the supporting objects are up

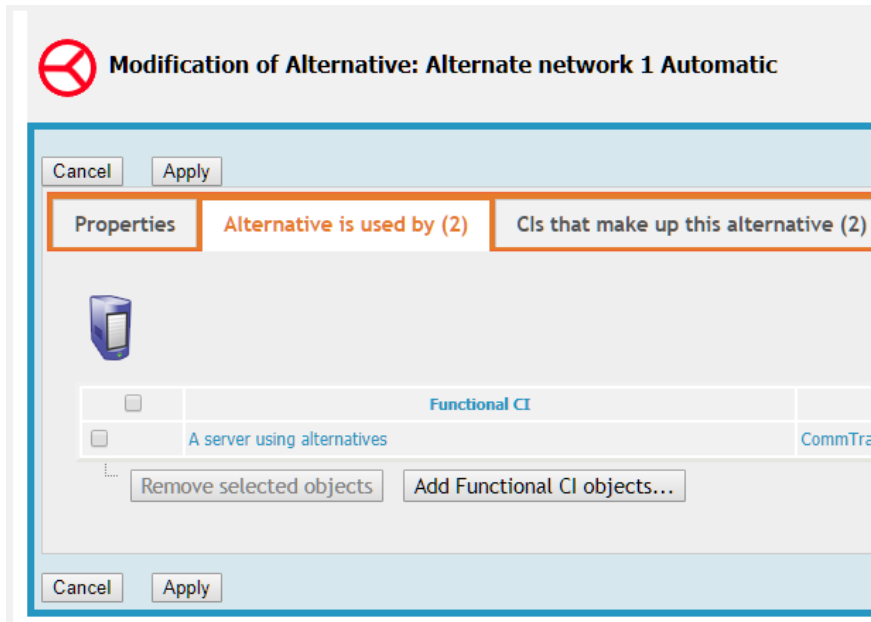
☒ The alternative is satisfied if at least 1 supporting objects is(are) up

☐ The alternative is satisfied if at least % of the supporting objects are up

Cancel Apply

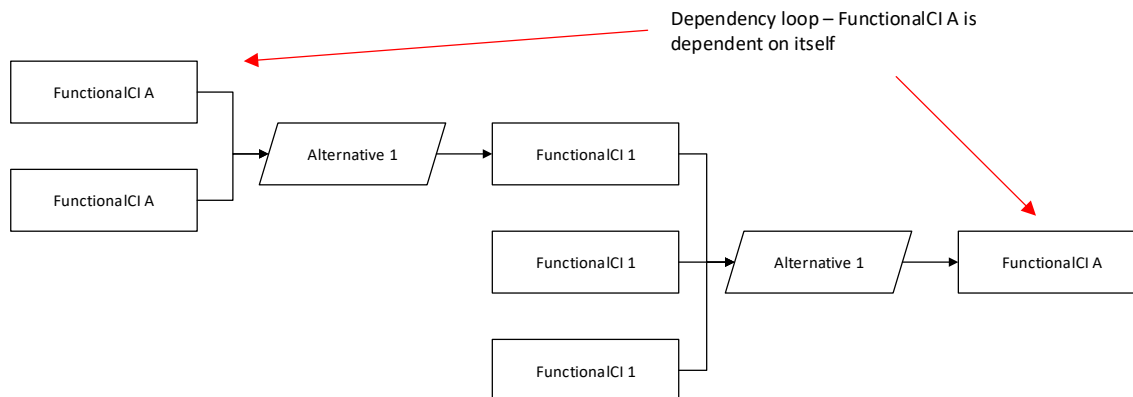
In this example, Network Device 395 and Network Device 396 are alternates to one another; only one has to be up for “Alternate network 1 Automatic” to be considered met.

4. Alternatives are used by target FunctionalCIs as dependencies, i.e. the target depends on the Alternatives.



In this example, Server “A server using alternatives” is using this alternative as one of its dependencies.

Be careful to avoid dependency loops; iTop apparently avoids an infinite loop but the results are really weird:



5. The target FunctionalCI has a tab that lets you specify which Alternatives are required for the target FunctionalCI to be considered “up” – all the Alternatives have to be up for the target FunctionalCI to be up.

The screenshot shows a web interface for a server configuration. At the top, there's a header bar with a server icon and the text 'Server: A server using alternatives'. To the right of the header are buttons for 'Modify...' and 'New...'. Below the header is a horizontal menu with tabs: 'Properties', 'Softwares', 'Contacts', 'Documents', 'Tickets', 'Network devices', 'Provider contracts', 'Services', 'Data received', 'Data sent', 'Used in alternative(s)', and 'Required alternative(s) (3)'. The 'Required alternative(s) (3)' tab is highlighted in orange, and a red arrow points to it. Below the tabs, there's a section with a red circle containing a white 'X' icon and the text 'Total: 3 objects.' Below this, there's a table with a single header row 'alternative id' and three data rows: 'Alternate servers 3 Automatic', 'Alternate network 1 Automatic', and 'Alternate SAN 50% Automatic'.

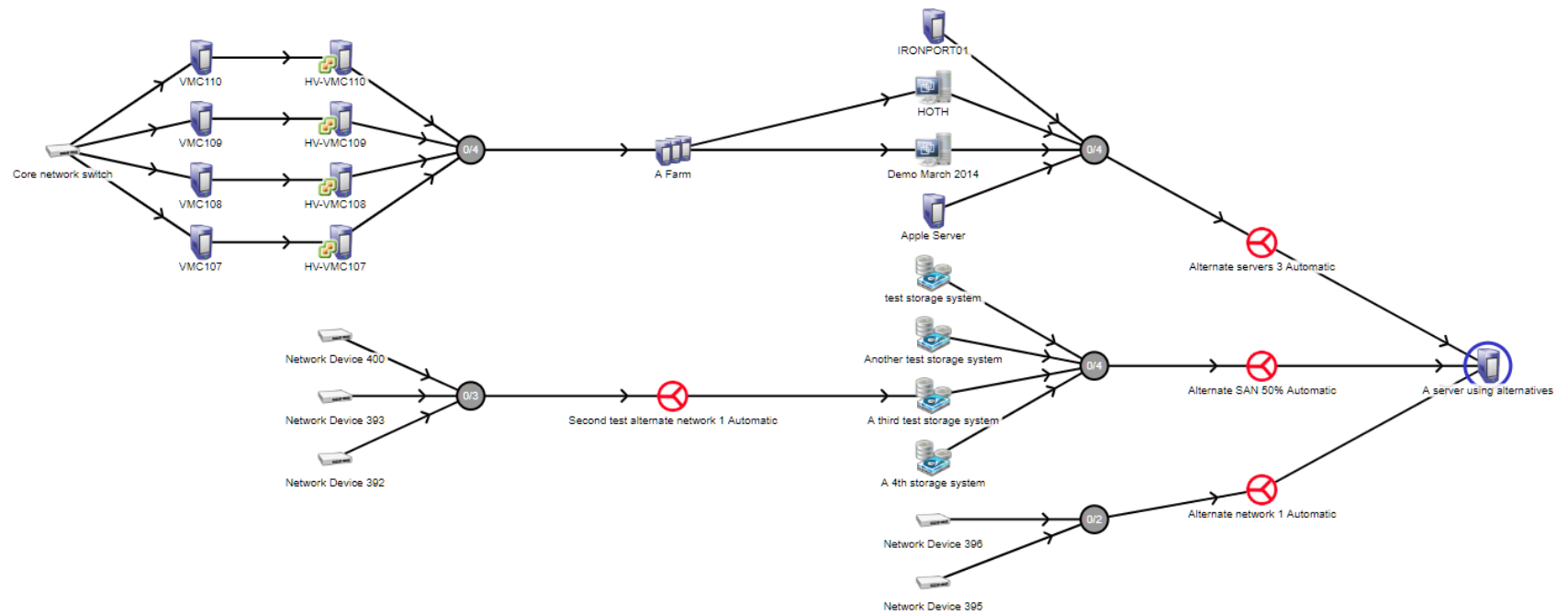
In this example, Server “A server using alternatives” has three Alternatives which have to be up in order for the Server to be up.

Please note that in order for this tab (and the Used in Alternatives tab) to be present, you must have added `alternative_list` and `requiredalternatives_list` to the `<detail>` presentation for the specific class as indicated in the installation instructions.

If Server “A server using alternatives” was itself used as a part of an Alternative, you could see that Alternative under the “Used in alternatives” tab on this page.

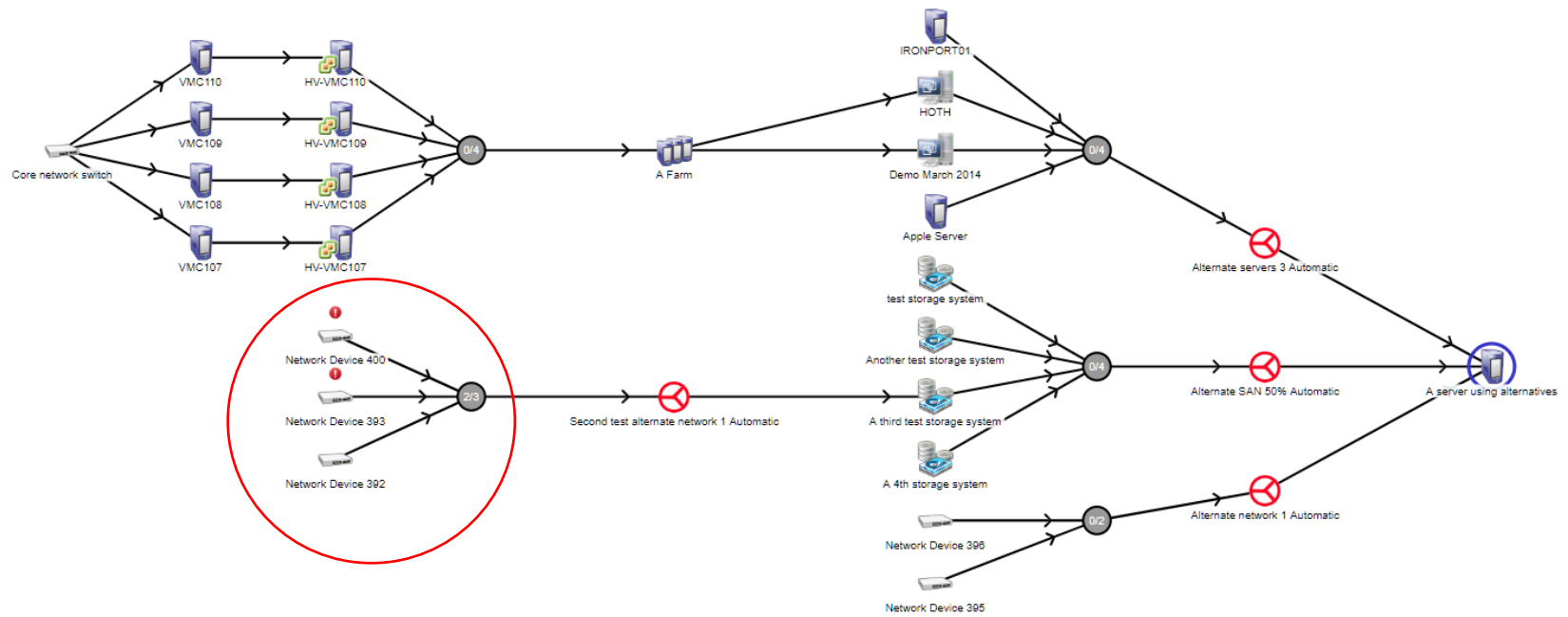


6. The Dependency diagram for Server “A server using alternatives” shows the full map including Alternatives:

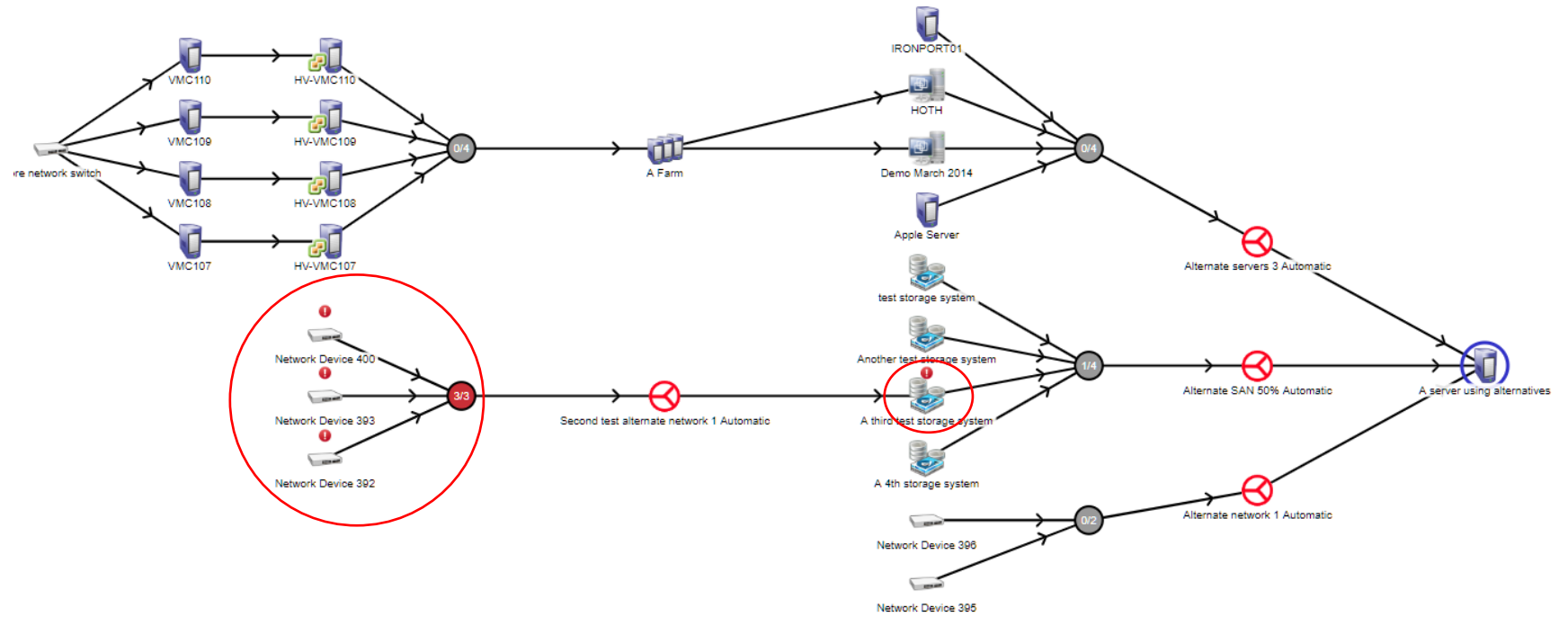


(Please note this diagram contains Alternatives in addition to the one created above).

7. Incidents on instances which are part of an Alternative but which don't trigger the redundancy condition of the Alternative don't affect higher level objects:



8. Incidents on instances which are part of an Alternative and which trigger the redundancy condition of the Alternative go to higher level tests:



9. But if the redundancy conditions for the Alternative are met then the incident flows through the Alternative and to higher-level tests. If an Alternative fails (as Alternate servers 3 Automatic has done here) then that failure goes to the FunctionalCI that is using the Alternative.

