Differences between TEC 3.8 and TEC 3.9 with respect to NetView 7.1.4

Jane Curry, Skills 1st Limited
jane.curry@skills-1st.co.uk

Abstract

This paper reviews the NetView / TEC integration capabilities that were available with NetView 7.1.3 and TEC 3.8. It then considers the new integration features added to each product with NetView 7.1.4 and TEC 3.9 respectively, and demonstrates solutions currently available out-of-the-box.

Although many NetView users seem to be upgrading to the latest version, upgrading TEC to the latest version appears to be moving far more slowly - it has far more implications; thus the latter half of the paper considers integration scenarios between NetView 7.1.4 and TEC 3.8.

Introduction

IBM Tivoli NetView and IBM Tivoli Enterprise Console have a long history of integration, dating back to the mid 1990s. Even before IBM bought Tivoli, a TEC adapter existed for NetView to provide a way of forwarding Simple Network Management Protocol (SNMP) TRAPs from a network management world, into the Tivoli problem management focal point known as TEC. This integration has grown closer over the years.

For some time now, integration has been two-way. Events can flow from NetView to TEC and TEC operators can bring up NetView consoles from their TEC Console interface. More recently, an event that is acknowledged or closed by a TEC operator will generate a similar

response automatically at the NetView system.

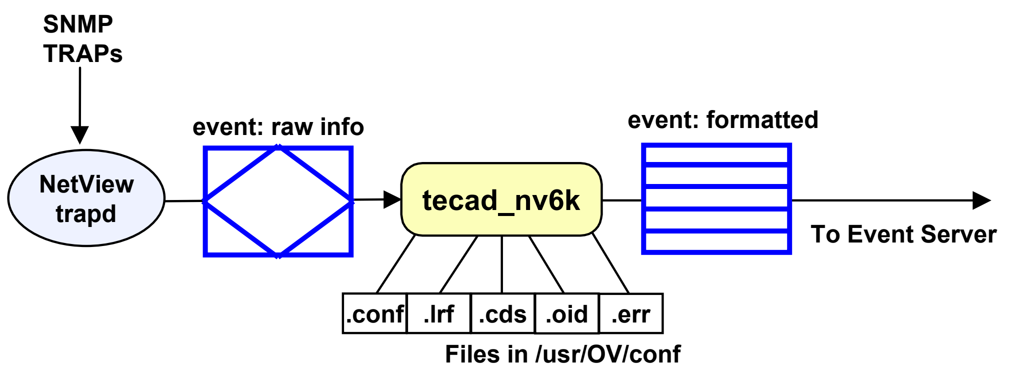
With NetView 7.1.4 and TEC 3.9, a further level of integration is being delivered that also incorporates IBM Tivoli Monitoring (ITM) and expands the scope of NetView beyond simple networking events into examining *services* that run on systems - currently the services that enjoy full support from NetView are DB2, WebSphere MQ and WebSphere Application Server (WAS).

Most of the comments in this paper apply regardless of the Operating System architecture that the NetView and TEC systems are deployed on. Some details *are* different if NetView is hosted on a Windows system and these differences will be explained.

NetView 7.1.3 / TEC 3.8 architecture

In the mid-1990s, a TEC adapter was shipped with the TEC product, to collect SNMP TRAPs from NetView on AIX, convert them into TEC event classes and forward those events to a TEC server. This TEC adapter was called tecad_nv6k.

Original TEC Adapter for NetView on Unix



The adapter took its events directly from NetView's trapd daemon, before NetView had the opportunity to do any analysis. Any configuration, including filtering of events, was done through standard Event Integration Facility (EIF) style TEC adapter configuration files:

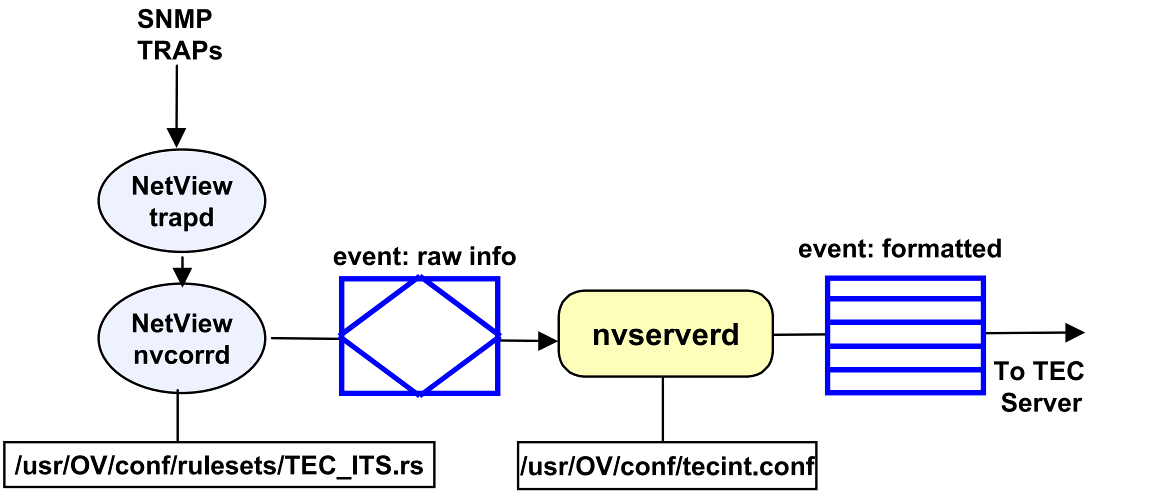
- .conf file — for configuring basic parameters such as TEC location, buffering
- also used to configure event filters
- .lrf file — **l**ocal **r**egistration **f**acility file to integrate tecad_nv6k with other NetView daemons
- .cds file — to specify mapping between SNMP TRAPs and TEC event classes
- .oid file — to map SNMP numeric Object Ids (OIDs) to names

- .err file — to control the level of error logging at the TEC adapter

The nvserverd TEC adapter for NetView on Unix

From NetView 5.1.2, a TEC adapter has been shipped with the NetView for Unix product. This **nvserverd** adapter is the preferred mechanism.

TEC Adapter for Unix NetView 5.2 - 7.1.3



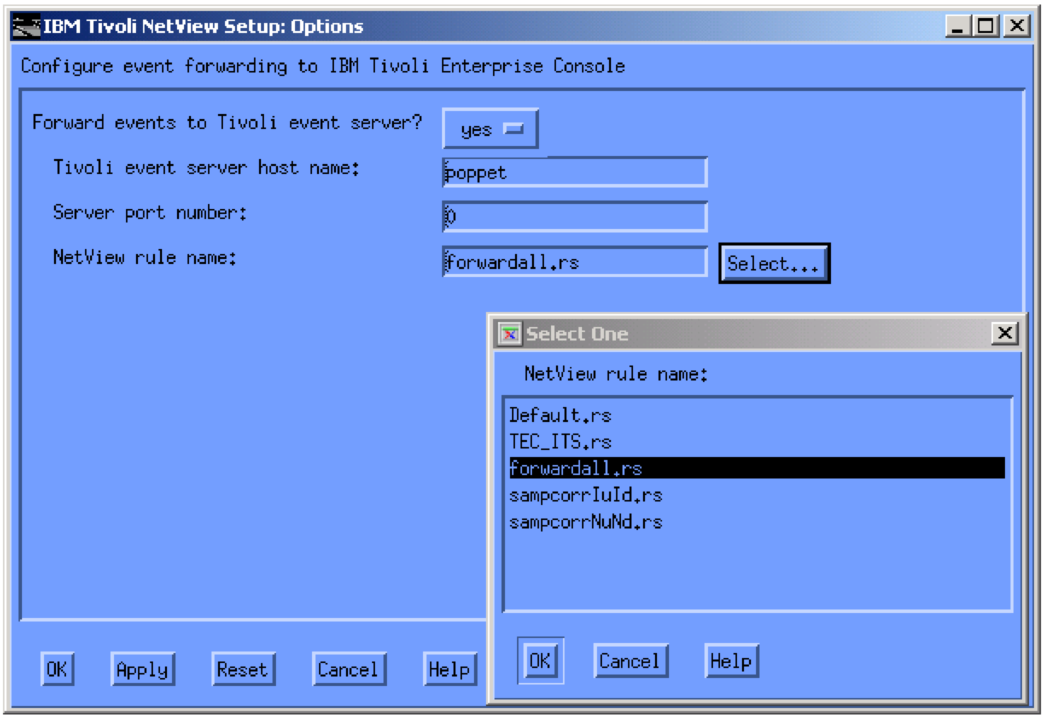
The nvserverd TEC adapter benefits from pre-processing by NetView's nvcorrd correlation engine. This means that SNMP events can be filtered out from the TEC stream by creating a NetView ruleset to specify which events are forwarded. This process is much simpler than configuring .oid and .cds files and a default NetView ruleset, **TEC_ITS.rs** has been shipped with the last few versions of NetView. The .conf file for nvserverd can be created either when NetView is installed, or through the **serversetup** utility, or it can be manually edited. The serversetup utility also configures:

- Whether the nvserverd adapter should forward events to TEC
- The location of the TEC server. This is either a resolvable hostname or an IP address - nvserverd is a non-TME TEC adapter
- The port used by the TEC server - this is 0 for a Unix TEC and 5529 (by default) for a Windows TEC
- The NetView ruleset to filter events in to the TEC

The configuration file, **/usr/OV/conf/tecint.conf**, can also be edited manually to add other standard EIF-style parameters, such as BufEvtPath for the location of the cache file. Changes to /usr/OV/conf/tecint.conf can be picked up by the nvserverd TEC adapter using the **nvtecia -reload** command. (**Note** that if you manually edit tecint.conf, make sure you take a copy of the file as any subsequent changes performed by serversetup will simply overwrite tecint.conf).

Configuring NetView for TEC Server

Serversetup -> Configure -> Configure event forwarding to IBM Tivoli Enterprise Console



IBM Tivoli NetView Setup: Options

Configure event forwarding to IBM Tivoli Enterprise Console

Forward events to Tivoli event server? yes

Tivoli event server host name: poppet

Server port number: 0

NetView rule name: forwardall.rs

Select One

NetView rule name:

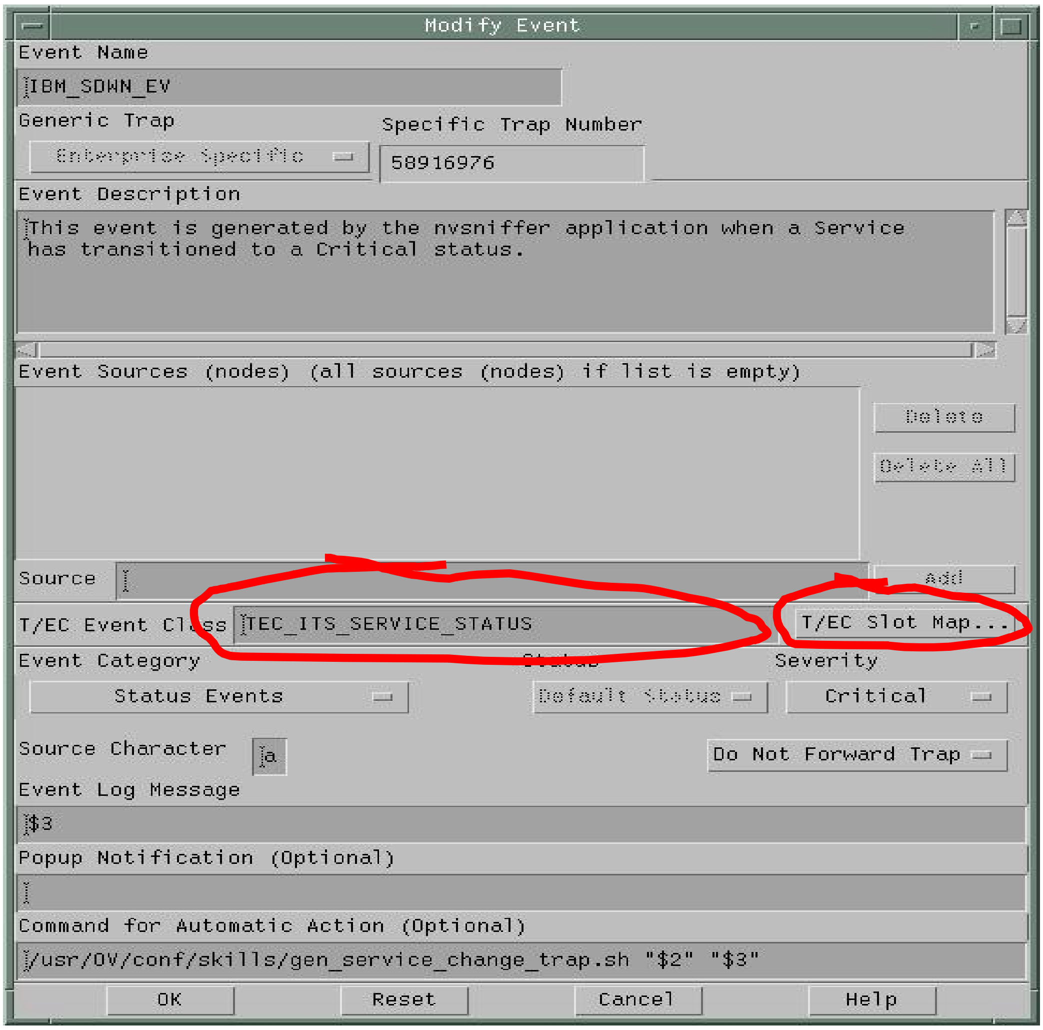
Default.rs
TEC_ITS.rs
forwardall.rs
sampcorrIuId.rs
sampcorrNuNd.rs

NetView on Unix comes with a large number of TRAPs pre-customized to be converted into TEC events. To extend this range is a very simple process using the menu utility provided to customize all other aspects of NetView TRAPs (Options -> Event Configuration -> Trap Customization). Parameter to be specified are:

- The TEC Class that this SNMP TRAP should be mapped to (these classes must be defined in the TEC Server in a baroc file incorporated into the active rulebase)

- Mapping of SNMP TRAP *variables* (varbinds) to TEC class *attributes*

Customizing nvserverd TRAPs



Modify Event

Event Name: IBM_SDWN_EV

Generic Trap: Enterprise Specific

Specific Trap Number: 58916976

Event Description: This event is generated by the nvsniffer application when a Service has transitioned to a Critical status.

Event Sources (nodes) (all sources (nodes) if list is empty)

Source:

T/EC Event Class: TEC_ITS_SERVICE_STATUS

Event Category: Status Events

Status: Default Status

Severity: Critical

Source Character: a

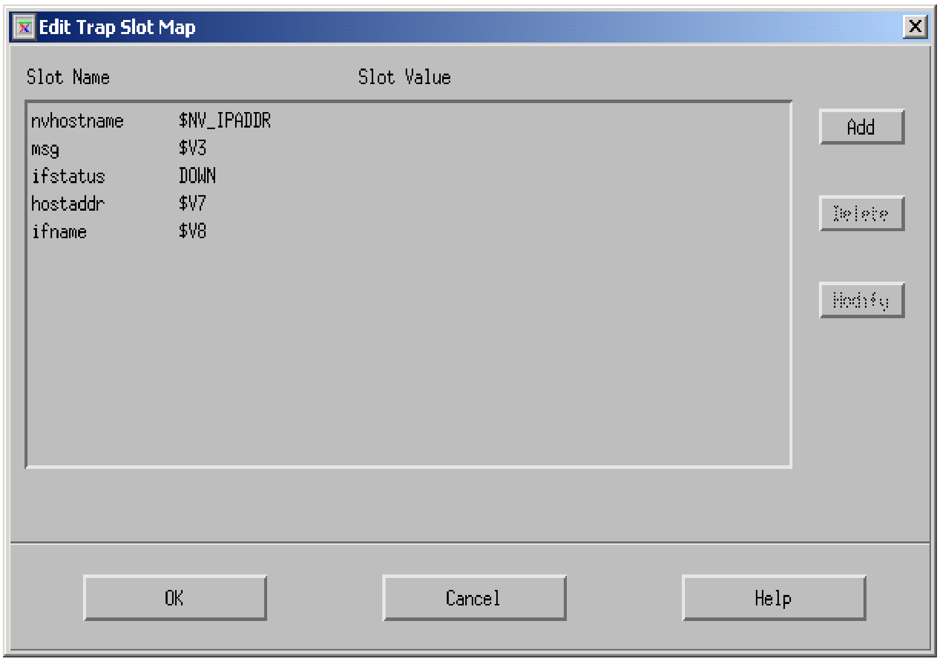
Do Not Forward Trap

Event Log Message: $3

Popup Notification (Optional):

Command for Automatic Action (Optional): /usr/OV/conf/skills/gen_service_change_trap.sh "$2" "$3"

Mapping TRAP Variables to TEC Slots



Slot Name	Slot Value
nvhostname	$NV_IPADDR
msg	$V3
ifstatus	DOWN
hostaddr	$V7
ifname	$V8

Any SNMP TRAP variable can be mapped to any legal TEC event class attribute. Source, sub_source, origin, and hostname are reserved and automatically included in the event (sub_source is populated with the single character NetView Category). The severity from the NetView customized event is automatically forwarded to the SEVERITY slot of the TEC event - the following mappings are used:

NetView Severity	**TEC Severity**
Cleared	HARMLESS
Indeterminate	UNKNOWN
Warning	WARNING
Minor	MINOR
Critical	CRITICAL
Major	FATAL

The superclass TEC_ITS_BASE event is defined with 15 variable attributes, named nv_var1 to nv_var15. These will be used to represent variables passed with a TRAP (Note that 15 is the limit to forward to TEC!). Make sure that when you specify attribute names, you are accurate. Case sensitivity is important!

The Slot Value field can have one of three types of value:

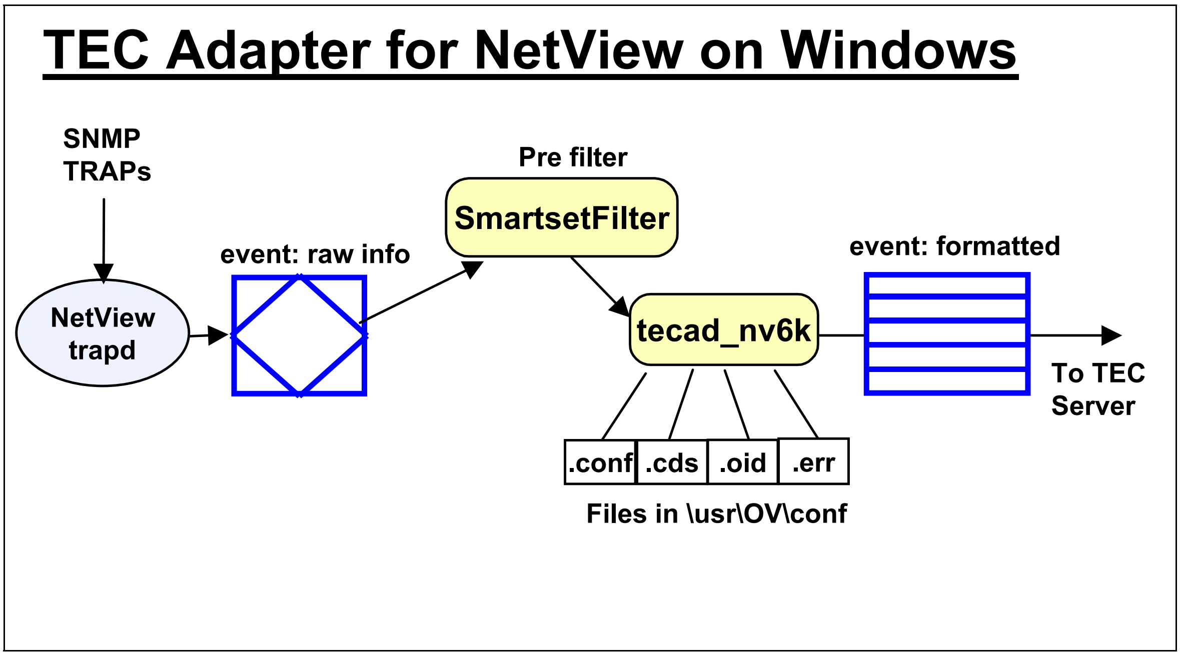
- Variable - This can be any trap variable specified as $V1, $V23 etc., where the number represents the positional variable on the incoming trap. The following variable names can also be used to pass trap data:

◆ $COMMUNITY	
◆ $ENTERPRISE	
◆ $SOURCE_TIME	
◆ $TYPE	i.e. 0 - 6 for trap type
◆ $SPECIFIC	
◆ $VARBIND	list of all non-fixed attributes

- Literal - This must be in double quotes.
- PRINTF statement - Example: PRINTF("As an example %s", $V1)

The tecad_nv6k TEC adapter for NetView on Windows

TEC integration is one area where NetView on Windows architecture varies greatly from NetView on Unix. The TEC forwarding mechanism does not uses nvcorrd and the Windows version of NetView does not have an nvserverd. The TEC adapter (rather confusingly) is also called tecad_nv6k and is shipped with NetView for Windows. It is an EIF-style adapter, configured using the same files as described above for the original tecad_nv6k (with the exception of the .lrf file).



TEC Adapter for NetView on Windows

To ease configuration, an application is supplied which uses menus to configure filters into tecad_nv6k.conf by selecting:

- SNMP TRAPs to be forwarded
- SmartSets of nodes for which the above TRAPs are to be forwarded

This application then generates appropriate filter statements in tecad_nv6k.conf. In addition to configuring filters, the TEC Server location, port number and communications mechanism (TME or non-TME) are also configured. The equivalent application can be run when NetView for Windows is installed, if such information is available at that time. A pre-filtering mechanism also exists by configuring \usr\OV\conf\tecSmartsetFilter.conf with SNMP Enterprise OID, TRAP numbers and SmartSets to be forwarded to the filters in tecad_nv6k.conf.

Although the Unix nvserverd has only been able to use non-TME communications to forward events to TEC (until version 7.1.4), tecad_nv6k on Windows has had the ability to use TME communications in earlier 7.x versions.

The difficulty with integrating NetView on Windows with TEC, is when new SNMP TRAPs need to be configured to be converted into TEC classes. tecad_nv6k.cds and tecad_nv6k.oid need to be manually edited (a fairly tedious, programming-style task), to specify the conversion from TRAP to event.

Integration features available for Unix and Windows NetView architectures

All NetView architectures have had the ability to convert SNMP TRAPs to TEC event classes and send them to a TEC server, for many years. Since TEC 3.7.1 Fixpack 2 with NetView 7.x, there have been a number of other integration points:

- An operator at a Java TEC Console can select an event and bring up a NetView Web Console, highlighting the indicated node, from the TEC Console menus
- TEC Consoles can have buttons customized for local procedures - a good example would be a button to **ping** the node highlighted in the TEC Console
- From a *NetView* native console a user can view *TEC* events for a selected node. This is dependent upon having the TME version of the Java TEC Console, installed on the operator's machine. It uses the TEC **wtdumper** command under-the-covers and the NetView **disptec** process (disptec has a log file at /usr/OV/log/disptec.log if this functionality doesn't produce the expected results).
- Utilities are provided to create TEC Event Groups relevant to NetView. The originally **collToEg** script created Event Groups based on NetView SmartSets. TEC 3.8 introduced **wcrtnvgroups** which generates event groups based on class and/or source.
-

Both Windows and Unix NetView architectures provide the **mib2trap** utility which parses any given SNMP MIB file for TRAP definitions. It then creates an output file containing NetView **addtrap** statements for each trap that it finds, using default customization parameters (such as for severity and category). This output file can be run to configure these new TRAPs into NetView's /usr/OV/conf/C/trapd.conf file. mib2trap can optionally produce baroc files if these TRAPs are likely to be forwarded to a TEC server. The full syntax for mib2trap is:

- mib2trap <MIB file> <output file> <baroc file> <base class>
- Note that if the <base class> parameter is omitted, the classes in the baroc file will all inherit from the **TEC_ITS_BASE** class.

Customizing the TEC Server for NetView

Class and rules files shipped for NetView

Over the years and the versions, a number of different TEC class definition files (.baroc files) and TEC ruleset files (.rls files) have been shipped with NetView TEC adapters:

- tecad_nv6k.baroc and ov_default.rls
- nvserverd.baroc and nvserverd.rls
- rfi.baroc and rfi.rls
- netview.baroc and netview.rls - these files have been part of the Default TEC rulebase since TEC 3.8

The current files for use in the rulebase on the TEC Server are **netview.baroc** and **netview.rls** (but be aware that these files have different *contents* in the TEC 3.8 Default rulebase and the TEC 3.9 Default rulebase). All NetView 7.1.3 and 7.1.4 TEC adapters now use the same TEC class structure for events, defined in netview.baroc.

TEC 3.8 classes for use with NetView

This section pertains strictly to the **netview.baroc** shipped with TEC 3.8. Be very aware that TEC 3.9 also ships a netview.baroc but it is *different*.

There are two classes defined in netview.baroc that inherit directly from the base event:

- TEC_ITS_BASE
- TEC_ITS_UNREACHABLE

TEC_ITS_BASE is a superclass from which most of the NetView events inherit. It:

- Redefines the severity default facet to be WARNING
- Introduces the following new attributes:
  - nvhostname: STRING;
  - networkid: STRING;
  - hostaddr: STRING;
  - category: TEC_ITS_CategoryE;
  - nv_enterprise: STRING;
  - nv_generic: INT32;
  - nv_specific: INT32;
  - nv_var1: STRING;
  - and 14 subsequent variables, nv_var2 through nv_var15

The TEC_ITS_UNREACHABLE class is used with the TEC 3.8 netview.rls and hb_ext.rls rulesets to correlate NetView subnet events with IBM Tivoli Monitoring (ITM) heartbeat events.

Note that TEC 3.8 provides a new rule primitive, **print_class_tree**, which is very useful for printing class hierarchies to an output file. An example is given below:

```
rule: print_classes: (
description: 'Fires on TEC_Start and prints NetView class hierarchy to
      /usr/OV/conf/skills',
event: _event of_class 'TEC_Start',
      reception_action: print_classes: (
            print_class_tree(
'/usr/OV/conf/skills/tec_netview_class.out', 'TEC_ITS_BASE' )
      )
).
```

The main subclasses under the TEC_ITS_BASE hierarchy are:

- TEC_ITS_BASE

- TEC_ITS_L3_STATUS
  - TEC_ITS_INTERFACE_STATUS
  - TEC_ITS_NODE_STATUS
  - TEC_ITS_SUBNET_CONNECTIVITY
  - TEC_ITS_ROUTER_STATUS
  - TEC_ITS_ISDN_STATUS
- TEC_ITS_OBJECT
- TEC_ITS_SERVER_EVENT
- TEC_ITS_SA_EVENT
  - TEC_ITS_SA_STATUS
  - TEC_ITS_L2_NODE_STATUS

There are a number of other leaf-class events that may be of interest:

- TEC_ITS_SNMPCOLLECT_THRESHOLD
- TEC_ITS_FORCED_POLL
- TEC_ITS_SNMP_STATUS_CHANGE

Note that event classes are included in netview.baroc to address IBM Tivoli Switch Analyzer (ITSA) events, and events where NetView has already performed correlation between layer 3 (NetView IP) events and layer 2 (ITSA) events.

netview.baroc makes extensive use of enumerated types for specifying attribute values. The main ones are:

• TEC_ITS_IfStatusE	UP/DOWN/ADMIN_DOWN/UNREACHABLE
• TEC_ITS_ NodeStatusE	UP/DOWN/ADMIN_DOWN/UNREACHABLE
• TEC_ITS_ RouterStatusE	UP/DOWN/MARGINAL/UNREACHABLE
• TEC_ITS_ IsdnStatusE	ACTIVE/DORMANT
• TEC_ITS_ SubnetConnE	UNREACHABLE/REACHABLE_AGAIN
• TEC_ITS_ SnmpColThrE	THRESHOLD_EXCEEDED/REARMED
• TEC_ITS_ NetworkStatusE	UP/DOWN/MARGINAL
• TEC_ITS_ MgmtStatusE	MANAGE/UNMANAGE
• TEC_ITS_ ActionE	ADDED/DELETED
• TEC_ITS_ L2StatusE	UP/DOWN/MARGINAL
• TEC_ITS_ SAStatusE	ifDown/nodeDown/nodeMarginal/ifUp /ifUnmanaged/ifDeleted/nodeUp/nodeUnmanaged/nodeResolved/nodeDeleted

Several of the leaf-node TEC classes in netview.baroc define extra attributes which are used in netview.rls. Here are the major ones:

- TEC_ITS_INTERFACE_STATUS
  - ifstatus: TEC_ITS_IfSTATUSE, default=UP;
  - ifname: STRING;
- TEC_ITS_ISDN_STATUS

  - isdnstatus: TEC_ITS_ISDN_STATUS, default=ACTIVE;
- TEC_ITS_SNMPCOLLECT_THRESHOLD
  - snmpstatus: TEC_ITS_SnmpColThrE, default=REARMED;
  - collection: STRING;
- TEC_ITS_NODE_STATUS
  - nodestatus: TEC_ITS_NodeStatusE, default=UP;
- TEC_ITS_ROUTER_STATUS
  - routerstatus: TEC_ITS_RouterStatusE, default=UP;
- TEC_ITS_SUBNET_CONNECTIVITY
  - reachability: TEC_ITS_SubnetConnE, default=REACHABLE_AGAIN;
  - subnetaddr: STRING;
  - subnetmask: STRING;
- TEC_ITS_INTERFACE_ADDED
  - action: TEC_ITS_ActionE, default=ADDED;
  - ifname: STRING;
- TEC_ITS_NODE_ADDED
  - action: TEC_ITS_ActionE, default=ADDED;
- TEC_ITS_INTERFACE_MANAGE
  - manage: TEC_ITS_MgmtStatusE, default=MANAGE;
  - ifname: STRING;
- TEC_ITS_NODE_MANAGE
  - manage: TEC_ITS_MgmtStatusE, default=MANAGE;
- TEC_ITS_SA_STATUS
  - sastatus: TEC_ITS_SAStatusE;
- TEC_ITS_L2_NODE_STATUS
  - l2status: TEC_ITS_L2StatusE, default=UP;
  - subnetaddr: STRING;
  - subnetmask: STRING;
- TEC_ITS_UNREACHABLE
  - ipunreachable: STRING;

TEC 3.8 rules for use with NetView

netview.rls is organized in a number of different sections:

- Initialization
  - Sets debug flag an filename
  - Sets latency variable for time to search back through event cache
  - Asserts Prolog predicates
- Severity adjustment rules
  - All NetView events default to a severity of WARNING from TEC_ITS_BASE
  - “Good news” events have severity set to HARMLESS
- Clearing rules

  - The clearing rules are all similar. An incoming event of a certain class clears **all** previous events of the same class, regardless of any attribute values (even attributes such as nodestatus, routerstatus, sastatus are not checked)
  - The clearing rules for Switch Analyzer events also check whether the incoming *and* previous SA events have an sastatus of ifDown, nodeDown or nodeMarginal; in this case the severity of the new event is increased to CRITICAL
- Synchronization between TEC and NetView
  - There are 2 basic types of synchronization rules to address **node** status and **interface** status
  - When a TEC Console operator changes the status of a TEC_ITS_INTERFACE_STATUS, TEC_ITS_NODE_STATUS, or TEC_ITS_ROUTER_STATUS event to **ACK**, **CLOSED** or from ACK to RESPONSE or OPEN (ie. an **UNACK**), then an SNMP TRAP is generated by TEC to the NetView server to ensure that the two management systems stay synchronized
  - An **ACK** change sends the NetView TRAP 50790450 - the Acknowledge TRAP. This should change the colour of the relevant device on the NetView map, to the dark green Acknowledge colour
  - An **UNACK** change sends the NetView TRAP 50790451 - the UnAcknowledge TRAP. This should change the colour of the relevant device on the NetView map, to the default colour (red if down, green if up)
  - A **CLOSED** change sends the NetView TRAP 50790412 - the Forced Poll TRAP, which has NetView immediately send a poll to the device, rather than wait for the next NetView polling period
  - Note that all these NetView TRAPs are configured as **LogOnly** by default in NetView. This means that they will appear in /usr/OV/log/trapd.log but will *not* appear in any user's event workspace
  - A number of Prolog predicates are **assert**ed to accomplish these TRAPs. The implementation is driven by a shellscript, $BINDIR/TME/TEC/nvsync.sh which calls a Java application
  - To improve performance, the rules will buffer upto 10 TRAPs or will buffer for upto 30 seconds (whichever criteria is reached first). These parameters can be customized.
- Root-cause correlation rules

  - The correlation rules are complex and are described in much more detail in the following section on TEC 3.9 rules. Appendix A has a complete table of all the rulesets in netview.rls, demonstrating the differences between the two versions.

The ruleset addresses events in the following categories:

- Layer 3 events from NetView itself
- Layer 2 events generated *by NetView* as a result of correlation between NetView and Switch Analyzer (TEC_ITS_L2_NODE_STATUS events)
- Layer 2 events that are generated by Switch Analyzer and forwarded by NetView (TEC_ITS_SA_STATUS events)

The *elements* that these events relate to are:

- Interfaces
- Nodes
- Routers
- Subnets

Here are some general premises that apply throughout netview.rls:

- In general, if there are node or router events and interface events from the *same* device then the root-cause is defined to be the **interface** event. (**Note** that this changes with the TEC 3.9 version of netview.rls!)
- In general, if there are Layer 3 events and Layer 2 events from the *same* device then the root-cause is defined to be the Layer 2 event (ie. a switch)
- The main NetView statuses that are considered are:
  - Up
  - Down
  - Admin Down
  - Marginal
  - Unreachable
- In general, non root-cause events have TEC status set to CLOSED, TEC severity set to HARMLESS and the event administrator attribute set to “netview.rls”. They are linked to causal events if possible.
- In general, duplicate TEC events have the TEC status of the *old* event set to CLOSED, the TEC severity of the *old* event set to HARMLESS and the event administrator attribute of the *old* event set to “netview.rls”
- “Duplicate” event tests generally do not use the dup_detect facet on attributes but do all_instances searches through the TEC rules cache, over a 10 minute period, checking for the same nvhostname attribute for the NetView that sent the event, and the same hostname attribute for the “problem” node. **Note** that NetView 7.1.3 Fixpack 1 and later versions ensure that the hostname attribute is either a Fully Qualified Domain Name (FQDN) (for preference), or an IP address. A short hostname will not be used. **Also note** that TEC 3.8 Fixpack 1 and later versions of TEC include the **fqhostname** attribute as part of the base event although this is not used by NetView unless the TEC 3.9 / NetView 7.1.4 State Correlation Engine (SCE) is used.

- If Unmanaged or Deleted events arrive for a device then all events for that device have status set to CLOSED, severity set to HARMLESS and administrator set to “netview.rls”
- TEC attributes will be populated throughout the various classes, as follows:

◆ adapter_host	FQDN of the NetView server
◆ nvhostname	IP address of the NetView server
◆ hostname	FQDN of the failing device if possible; otherwise IP address
◆ origin	IP address of failing device

The logic of netview.rls will not be discussed in detail here; rather, it will be compared to the netview.rls shipped with TEC 3.9, in the section on TEC 3.9. Appendix A is a table of all the rules in TEC 3.8 / 3.9 showing where 3.8 differs.

NetView 7.1.4 / TEC 3.9 architecture

NetView 7.1.4 and TEC 3.9 were released in October 2003. Many of the enhancements are related to greater integration between TEC and NetView and also greater integration with IBM Tivoli Monitoring (ITM) 5.1.1.

NetView 7.1.4 enhancements

- servmon daemon to replace nvsniffer for monitoring ports, services and applications
- itmquery utility to allow NetView to query ITM information for Tivoli Endpoints, either via command-line or from servmon
- nvserverd TEC adapter can now use either non-TME communications or TME communications
- nvserverd TEC adapter now integrates with a local TEC State Correlation Engine (SCE), by default. This capability can be turned off in tecint.conf, if desired
- In

TEC 3.9 enhancements

- New Web Console for TEC based on WebSphere Application Server (WAS) 5.0 Base Edition
- TEC Gateway can now receive non-TME events via the tec_gwr process
- Enhancements to state correlation including:
  - Correlation of events from TME adapters with events from non-TME adapters at the Tivoli Enterprise Console gateway
  - Support for customizable actions
  - New state correlation rules

- New Default rule base, which includes preconfigured rule sets that provide support for processing common application and infrastructure events. The rules in the Default rule set provide functions that include the following:

  - Causal analysis of network infrastructure and e-business application events based on service impact and dependency relationships
  - Scheduling of maintenance windows and discarding of events from systems currently undergoing maintenance
  - Integration with external trouble ticket systems
  - Heartbeat monitoring and detection of missed heartbeat pulses
- Changes to netview.baroc and netview.rls to provide greater integration between TEC, NetView and ITM
- TEC 3.9 root.baroc has fqhostname attribute (in fact, this was introduced with TEC 3.8 Fixpack 1)
- New compiled Prolog predicates provided in Default rulebase TEC_TEMPLATES, including predicates for tracing and logging (tracelog.wic), maintenance

(maintenance.wic), notification (notify.wic), troubleticketing (troubleticket.wic), dependencies (dependency.wic) and e-business (ebusiness.wic)
- New manual "TEC 3.9 Ruleset Reference" which documents all rules in all rulesets in the Default rulebase
- Greater integration with Tivoli Data Warehouse
- New installation wizard CD

netview.baroc in TEC 3.9

It is extremely important to understand that the *name* of the NetView baroc and ruleset files have not changed between TEC 3.8 and TEC 3.9; however the contents *have* changed.

The main differences are to incorporate service events generated by the State Correlation Engine behind nvserverd and to provide correlation between these events and those coming from ITM. The following new classes have been added (only the subnet class has extra attributes):

- TEC_ITS_SERVICE_IMPACT ISA TEC_ITS_BASE
- TEC_ITS_NODE_SERVICE_IMPACT ISA TEC_ITS_SERVICE_IMPACT
- TEC_ITS_SUBNET_SERVICE_IMPACT ISA TEC_ITS_SERVICE_IMPACT
  - subnetaddr: STRING;
  - nodelist: LIST_OF STRING, default = [];

TEC_ITS_BASE has been modified slightly to set default facets:

- source: default = 'ITS';
- sub_source: default = 'N/A';
- category: default = 'undefined';

The enumeration type, TEC_ITS_CategoryE now has 3 extra values:

- 0 "undefined"
- 98 "sniffer"
- 99 "itm"

The default severity facet on the TEC_ITS_UNREACHABLE event is changed from FATAL (in TEC 3.8) to WARNING in TEC 3.9

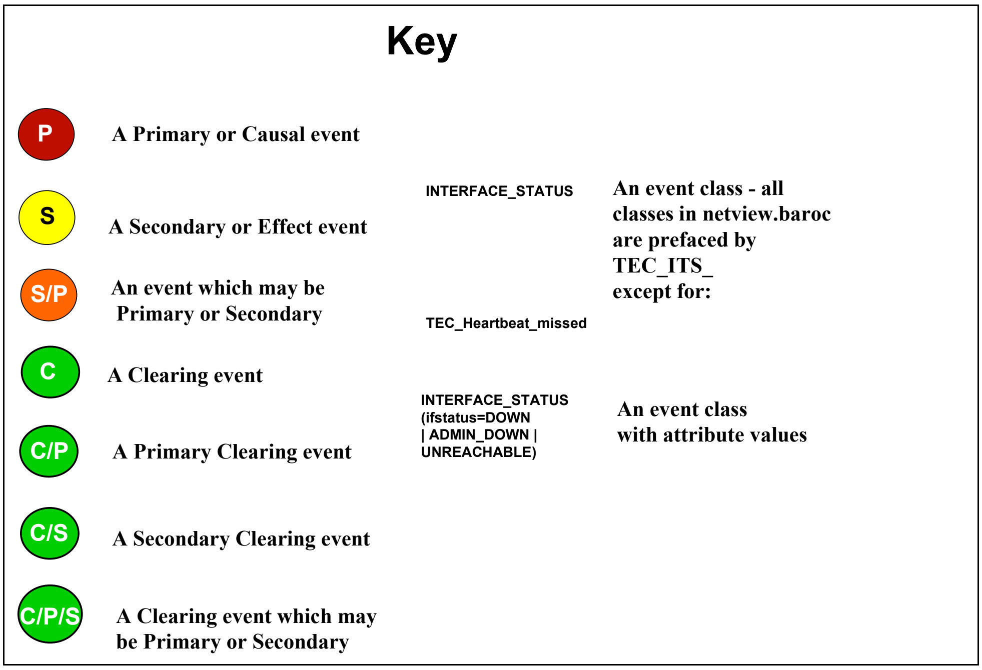
netview.rls in TEC 3.9

It is extremely important to understand that the *name* of the NetView baroc and ruleset files have not changed between TEC 3.8 and TEC 3.9; however the contents *have* changed.

Many of the rules within netview.rls are identical in the TEC 3.8 and TEC 3.9 versions. New rules and additions to existing rules have been engineered to accommodate service events and their correlation with other NetView and ITM events.

There is one **huge** difference in the logic between TEC 3.8 and TEC 3.9. TEC 3.8 used the basic premise that an **Interface** event was a **causal** event and node / router events would be **effect** events. TEC 3.9 changes that logic such that the **causal** event is always defined to be the **node** or **router** event. The only exception is where a TEC_ITS_ROUTER_STATUS event with the routerstatus attribute set to MARGINAL or UNREACHABLE (ie. not DOWN), will be an effect event of a related TEC_ITS_INTERFACE_STATUS event.

The following diagrams portray the logic of the correlation found in the TEC 3.9 rules. The key for the diagrams is as follows:


Key
P
A Primary or Causal event
S
A Secondary or Effect event
S/P
An event which may be Primary or Secondary
C
A Clearing event
C/P
A Primary Clearing event
C/S
A Secondary Clearing event
C/P/S
A Clearing event which may be Primary or Secondary
INTERFACE_STATUS
An event class - all classes in netview.baroc are prefaced by TEC_ITS_ except for:
TEC_Heartbeat_missed
INTERFACE_STATUS (ifstatus=DOWN | ADMIN_DOWN | UNREACHABLE)
An event class with attribute values

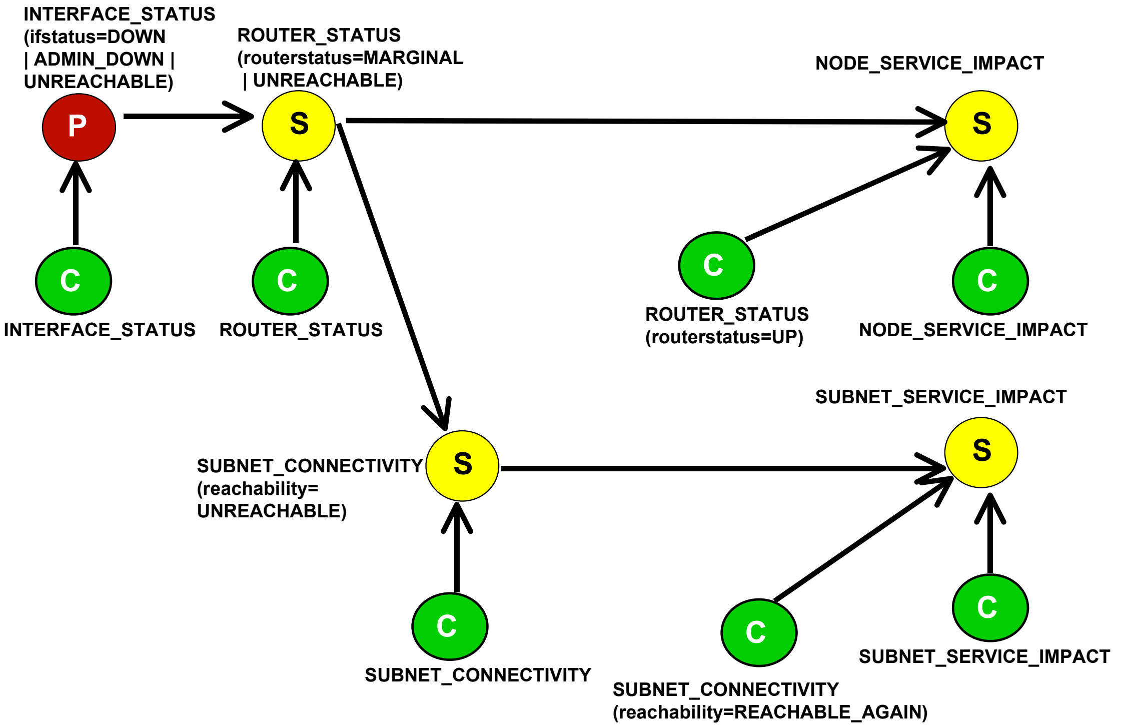
If a router loses one or more interfaces, but not all interfaces, NetView will generate the relevant number of Interface Down TRAPs and a Router Marginal TRAP. As a result, some networks may also become unreachable, with resulting Network Unreachable TRAPs. The Router Fault Isolation (RFI) algorithm within NetView's netmon process determines whether networks have become unreachable as a result of losing an interface, or whether there is another path to the network. This scenario is the only one in the TEC 3.9 ruleset where an **Interface** is a root-cause.

A Network Unreachable TRAP is converted to a TEC_ITS_SUBNET_CONNECTIVITY event with a reachability attribute of UNREACHABLE. This event may also be a causal event of a TEC_ITS_SUBNET_SERVICE_IMPACT event that may have been generated when NetView's servmon daemon detects that, say, a DB2 service is no longer available. The TEC_ITS_SUBNET_SERVICE_IMPACT event is actually generated by the NetView TEC State Correlation Engine (SCE) behind nvserverd.

The failing router may not be strictly a networking device, such as a Cisco router, but may be a computer acting as a router, also running other services. If NetView's servmon daemon is monitoring these services, a TEC_ITS_NODE_SERVICE_IMPACT event may be generated as a result of the degradation of the device.

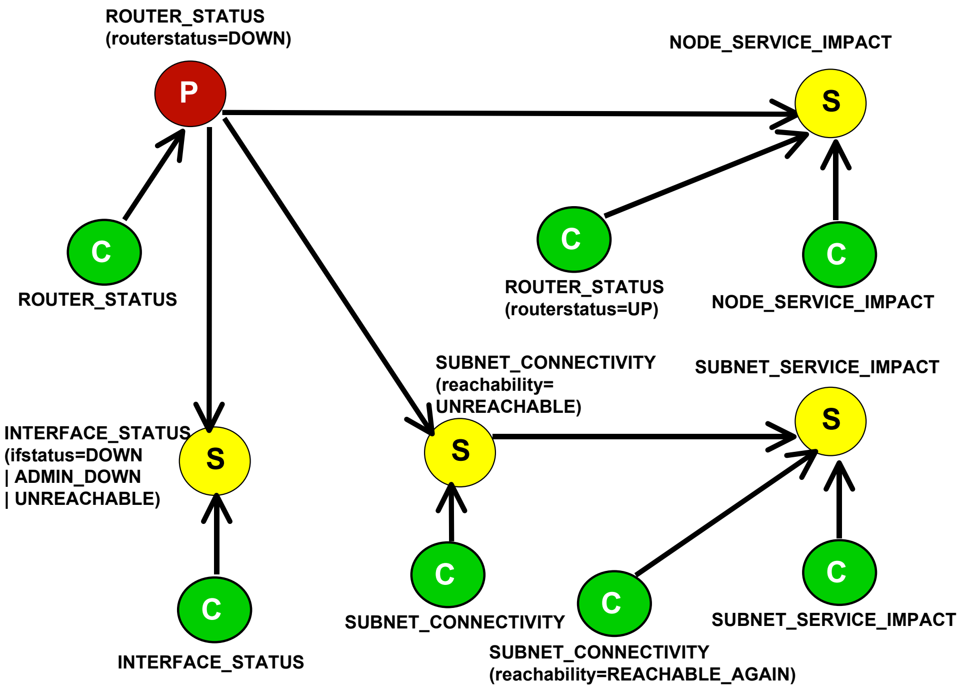
The clearing section of rules provides for any event of a certain class to clear all other events of the same class.

Interface(s) Down as root cause



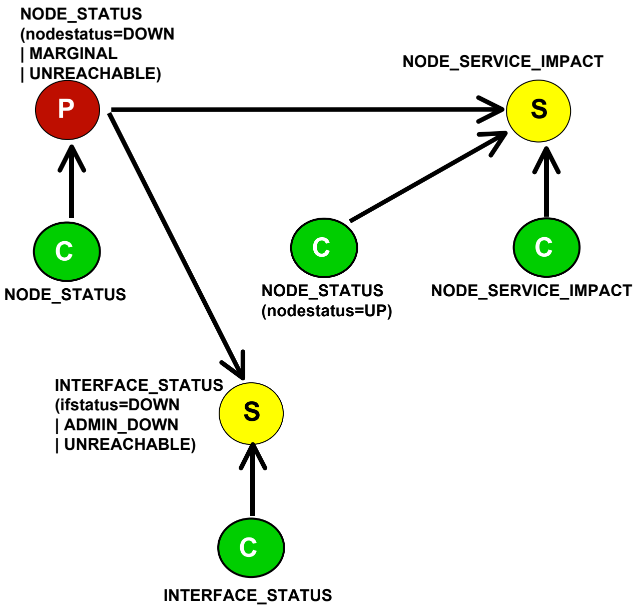
If a router fails totally, NetView will generate the requisite number of Interface Down events, followed by a Router Down event; there may also be a Router Marginal event in between. In this case, the TEC_ITS_ROUTER_STATUS event is the root cause of potentially a great number of effect events.

Router Down as root cause

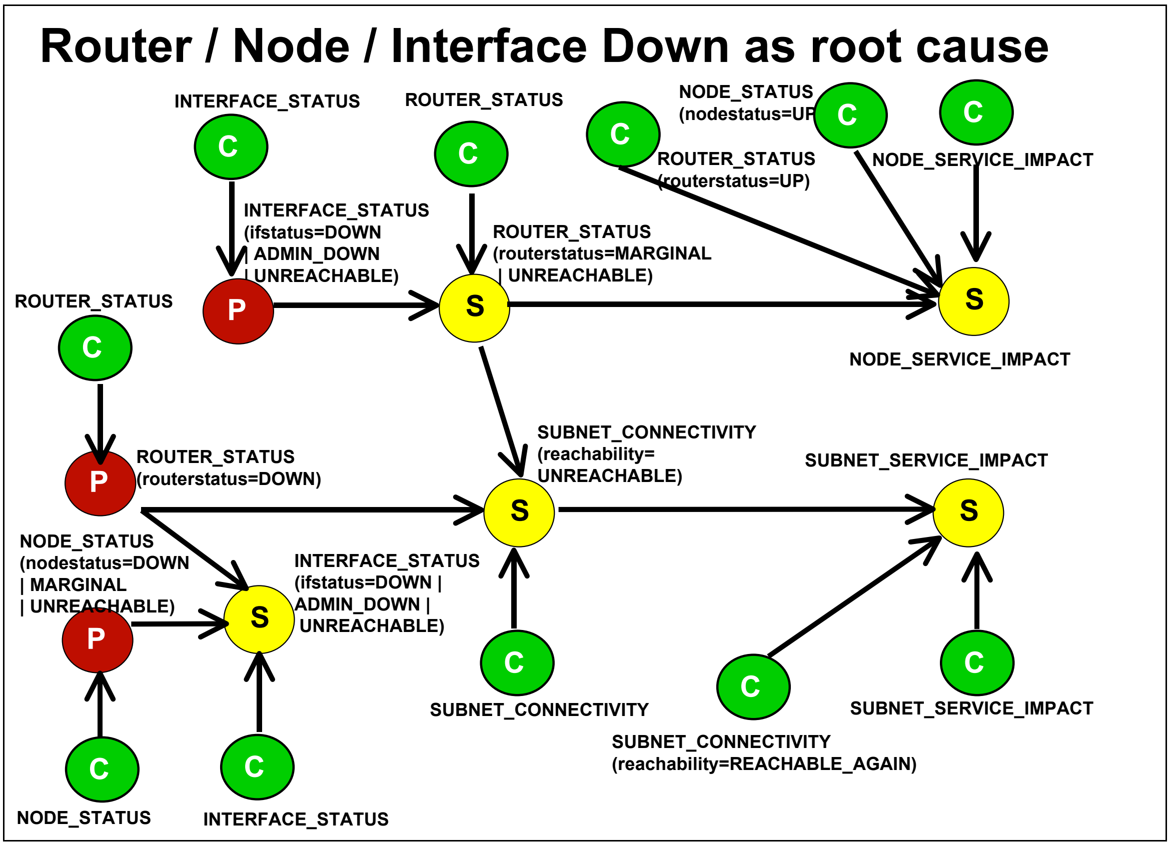


If NetView detects a Node going down, again the node event is the primary, causal event and the interface event is the secondary or effect event. In this case, there are no subnet complications but there may be further effect events if NetView is monitoring services on the node.

Node Down as root cause



Putting together these three scenarios gives the following logic diagram:

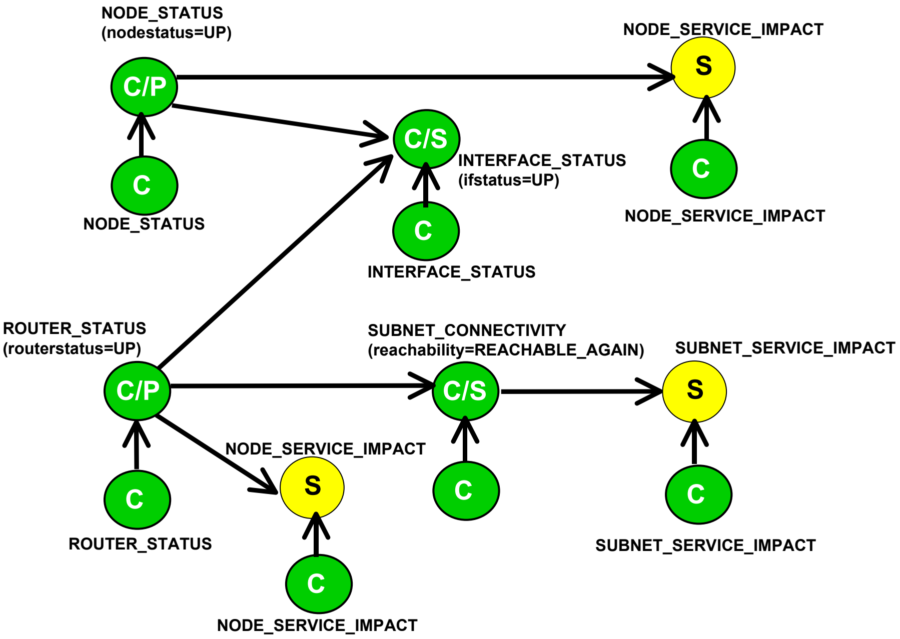


Extra modifications in the clearing section of netview.rls with TEC 3.9, provide for “good news” events to clear service events:

- TEC_ITS_NODE_STATUS with attribute nodestatus=UP clears TEC_ITS_NODE_SERVICE_IMPACT events
- TEC_ITS_ROUTER_STATUS with attribute routerstatus=UP clears TEC_ITS_NODE_SERVICE_IMPACT events
- TEC_ITS_SUBNET_CONNECTIVITY with attribute reachability=REACHABLE_AGAIN clears TEC_ITS_SUBNET_SERVICE_IMPACT events

The complete picture for the clearing event logic is shown below:

Cause / effect clearing events



NODE_STATUS (nodestatus=UP)

NODE_SERVICE_IMPACT

INTERFACE_STATUS (ifstatus=UP)

NODE_STATUS

NODE_SERVICE_IMPACT

INTERFACE_STATUS

ROUTER_STATUS (routerstatus=UP)

SUBNET_CONNECTIVITY (reachability=REACHABLE_AGAIN)

SUBNET_SERVICE_IMPACT

NODE_SERVICE_IMPACT

ROUTER_STATUS

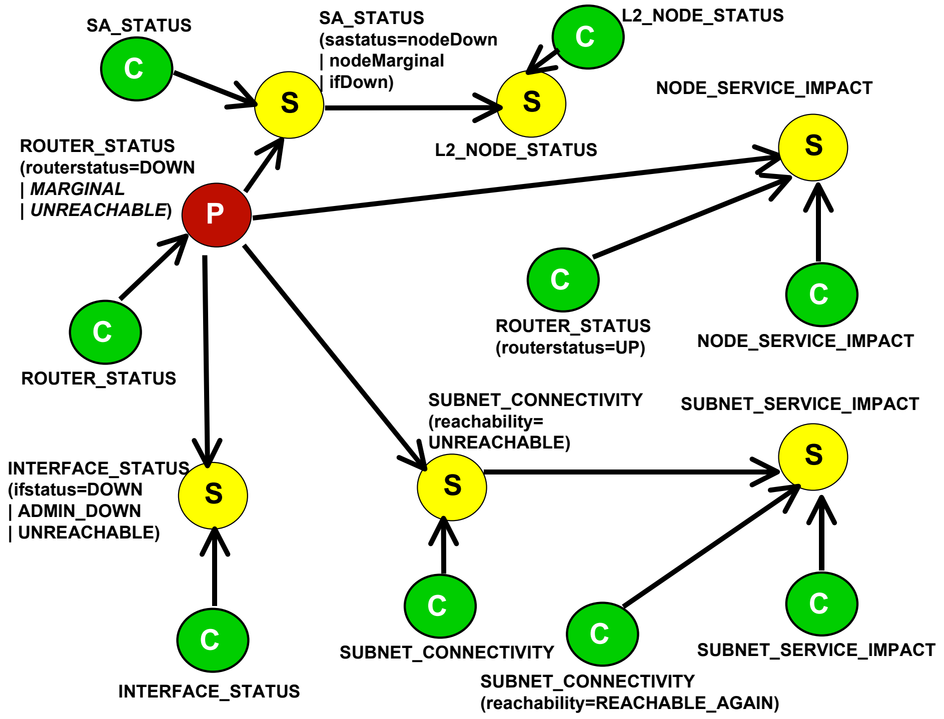
SUBNET_SERVICE_IMPACT

NODE_SERVICE_IMPACT

The netview.rls ruleset has always addressed Switch Analyzer events as well as IP events. A router event is always the primary causal event and TEC_ITS_SA- style events are effects.

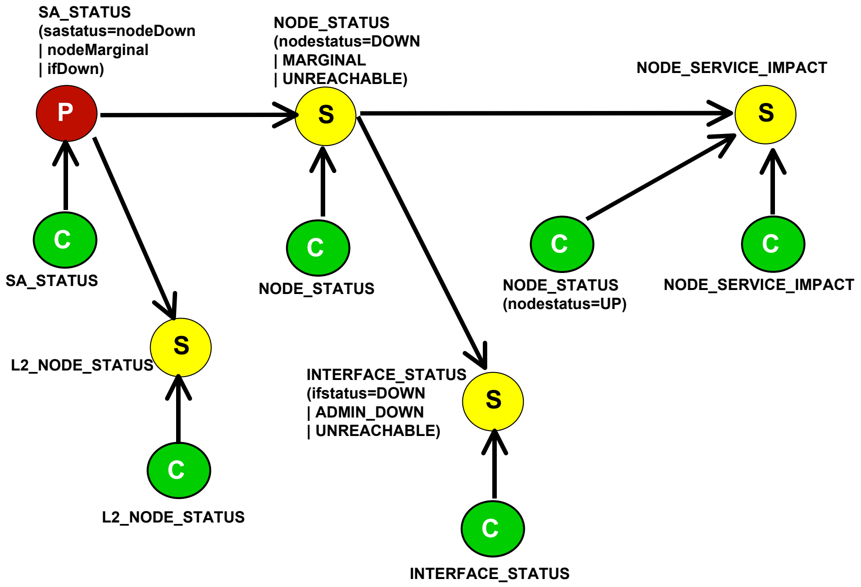
In addition, if NetView has generated layer 2 events, TEC will correlate these as effect events of Switch Analyzer events.

Router down causes SA; SA causes L2

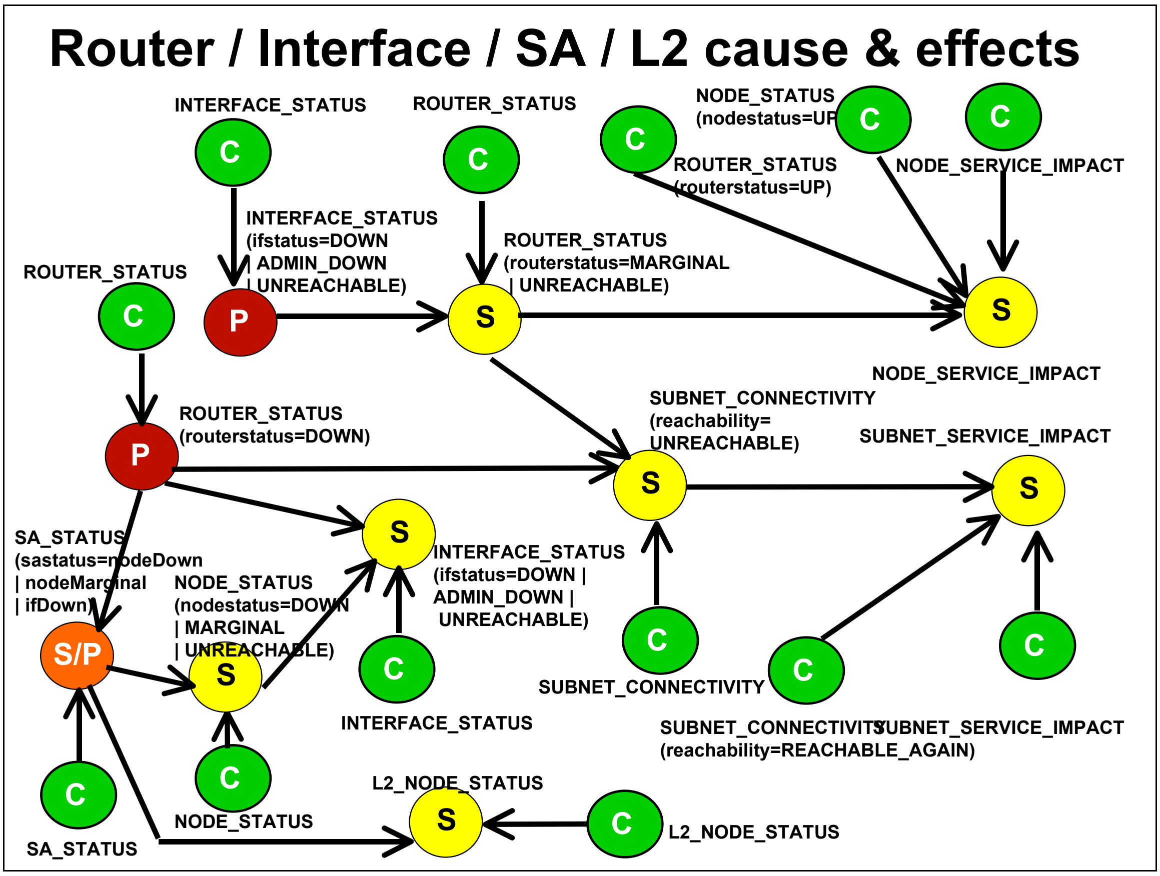


On the other hand, if a node down event is received, then the root cause will be correlated to be the Switch Analyzer event, rather than the node event. Any L2 events again are effects.

Switch Analyzer as root of node & L2 events



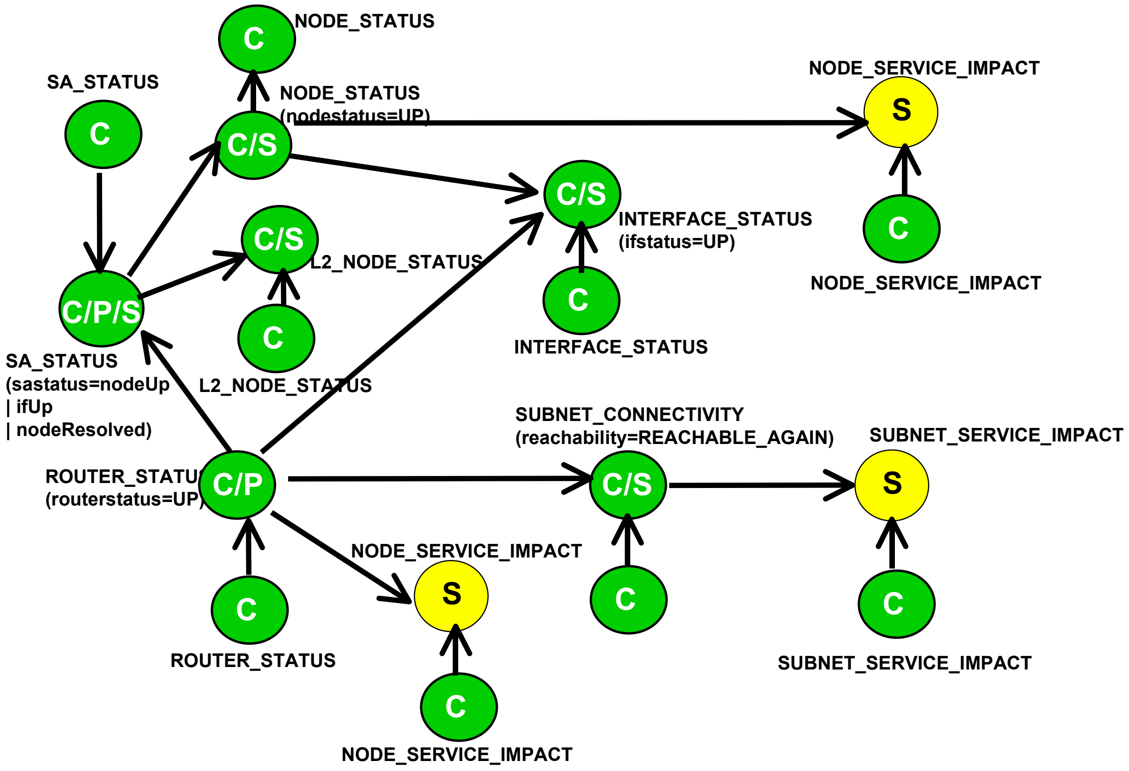
Putting all these scenarios and rule logic together gives:



Router / Interface / SA / L2 cause & effects

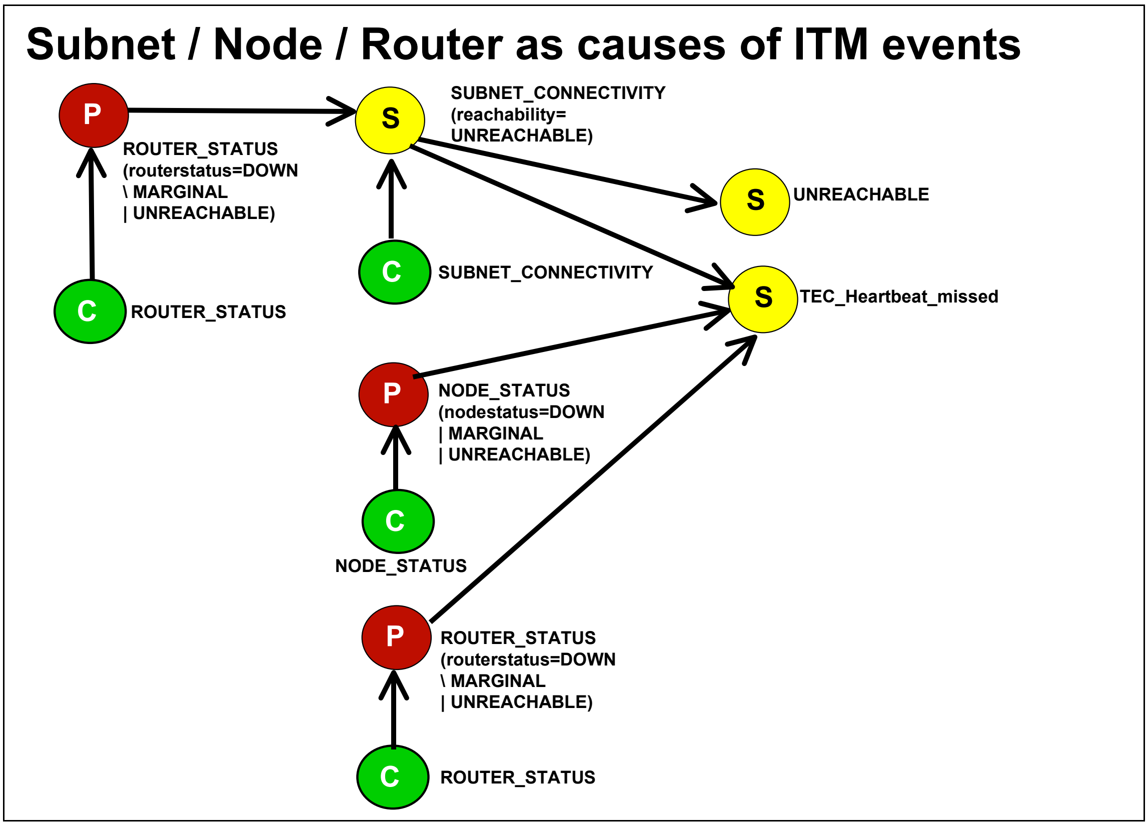
The full picture for clearing events is:

Cause / effect clearing events with SA



The TEC 3.8 ruleset correlated TEC_ITS_UNREACHABLE events generated by ITM, with TEC_ITS_SUBNET_CONNECTIVITY events. This means that a multitude of ITM effect events can all be correlated to one root-cause event of a subnet becoming unreachable. The TEC 3.9 ruleset extends this logic to also correlate TEC_Heartbeat_missing ITM effect events to a causal TEC_ITS_SUBNET_CONNECTIVITY, TEC_ITS_ROUTER_STATUS or TEC_ITS_NODE_STATUS event.

When looking for “duplicate” events, this last category of correlation rules uses the fqhostname attribute of events for comparison. If the NetView event has not populated the fqhostname attribute then the hostname attribute of the NetView event is compared against the fqhostname attribute on the ITM event.



Subnet / Node / Router as causes of ITM events

Summary

The purpose of this document was to examine the integration between NetView and TEC, highlighting the differences between NetView 7.1.3 with TEC 3.8, and NetView 7.1.4 with TEC 3.9.

The netview.rls ruleset has been examined in some detail to understand the integration between NetView, TEC and ITM.

The team that wrote this paper

Jane Curry is an independent Tivoli consultant and instructor, specializing in the Tivoli availability products - Framework, NetView, TEC and ITM. Previously, she spent 11 years in IBM working in both presales and postsales, systems and network management roles.

Thanks are due to IBM for permission to copy and use the netview.rls section of the TEC 3.9 Ruleset Reference manual.

Appendix A Rules in the netview.rls ruleset in TEC 3.9 and TEC 3.8

The text for the TEC 3.9 netview.rls ruleset has been reproduced from the “TEC 3.9 Ruleset Reference” manual, pages 43 - 58, with permission from the International Business Machines Corporation.

Startup and Shutdown rules

Rule Name	TEC 3.8	TEC 3.9

netview_configure	Initialisation rule is called **startup**
	Defaults are:
	netview_admin = ‘netview.rls’
	latency = 600 seconds
	netview_debug = ‘no’
	netview_logfile = ‘netview.log’ (in $DBDIR if not absolute)
	stdout = ‘netview.out’ (in $DBDIR if not absolute)
	stderr = ‘netview.err’ (in $DBDIR if not absolute)
	Predicates asserted for logging and debugging - logstr, logfmt, logevt, debug_msg. (TEC 3.9 uses new predicates defined in TEC_TEMPLATES/tracelog.wic). Also 4 predicates to check if a binary IP address is contained in a binary subnet address/mask combination.
		The netview_configure rule is a configuration rule that runs upon receipt of the TEC_Start event, which is sent during intialization of the event server. This rule sets global parameters for the NetView rules, asserts auxiliary predicates for internal use, and initializes the log files for the rule set. By customizing this rule, you can configure the behavior of the netview.rls rule set.
		
		Defaults are:
		netview_admin = ‘netview.rls’
		nv_latency = 600 seconds
		nvsync_timeout = 30 seconds
		nvsync_port = 162
		nvsync_maxhosts = 10
		netview_debug = ‘no’
		netview_logfile = ‘netview.log’ (in $DBDIR if not absolute)

shutdown		The shutdown rule runs upon receipt of the TEC_Stop event, which is sent when the event server shuts down. This rule finalizes any open log files for the rule set.

Severity adjustment rules

Rule Name	TEC 3.8	TEC 3.9
router_raise	Does not exist in TEC 3.8 netview.rls	The router_raise rule runs upon receipt of the

Rule Name	**TEC 3.8**	**TEC 3.9**
		TEC_ITS_ROUTER_STATUS event. If the routerstatus attribute of the received event is equal to DOWN, the severity of the event is set to CRITICAL.
interface_lower	TEC 3.8 rule same as TEC 3.9	The interface_lower rule runs upon receipt of the TEC_ITS_INTERFACE_STATUS event. If the ifstatus attribute of the received event is equal to UP, the severity of the event is set to HARMLESS.
isdn_lower	TEC 3.8 rule same as TEC 3.9	The isdn_lower rule runs upon receipt of the TEC_ITS_ISDN_STATUS event. If the isdnstatus attribute of the received event is equal to ACTIVE, the severity of the event is set to HARMLESS.
snmp_lower	TEC 3.8 rule same as TEC 3.9	The snmp_lower rule runs upon receipt of the TEC_ITS_SNMPCOLLECT_THRESHOLD event. If the snmpstatus attribute of the received event is equal to REARMED, the severity of the event is set to HARMLESS.
node_lower	TEC 3.8 rule same as TEC 3.9	The node_lower rule runs upon receipt of the TEC_ITS_NODE_STATUS event. If the nodestatus attribute of the received event is equal to UP, the severity of the event is set to HARMLESS.
router_lower	TEC 3.8 rule same as TEC 3.9	The router_lower rule runs upon receipt of the TEC_ITS_ROUTER_STATUS event. If the routerstatus attribute of the received event is equal to UP, the severity of the event is set to HARMLESS.
subnet_lower	TEC 3.8 rule same as TEC 3.9	The subnet_lower rule runs upon receipt of the TEC_ITS_SUBNET_CONNECTIVITY event. If the reachability attribute of the received event is equal to REACHABLE_AGAIN, the severity of the event is set to HARMLESS.
interface_added_lower	TEC 3.8 rule same as TEC 3.9	The interface_added_lower rule runs upon receipt of the TEC_ITS_INTERFACE_ADDED event. If the action attribute of the received event is equal to ADDED, the severity of the event is set to HARMLESS.
interface_managed_lower	TEC 3.8 rule same as TEC 3.9	The interface_managed_lower rule runs upon receipt of the TEC_ITS_INTERFACE_MANAGE event. If the manage attribute of the received event is equal to MANAGE, the severity of the event is set to HARMLESS.

Rule Name	TEC 3.8	TEC 3.9
node_added_lower	TEC 3.8 rule same as TEC 3.9	The node_added_lower rule runs upon receipt of the TEC_ITS_NODE_ADDED event. If the action attribute of the received event is equal to ADDED, the severity of the event is set to HARMLESS.
node_managed_lower	TEC 3.8 rule same as TEC 3.9	The node_managed_lower rule runs upon receipt of the TEC_ITS_NODE_MANAGE event. If the manage attribute of the received event is equal to MANAGE, the severity of the event is set to HARMLESS.
sa_status_lower	TEC 3.8 rule same as TEC 3.9	The sa_status_lower rule runs upon receipt of the TEC_ITS_SA_STATUS event. If the sastatus attribute of the received event is equal to ifUp, nodeUp, or nodeResolved, the severity of the event is set to HARMLESS.
l2_status_lower	TEC 3.8 rule same as TEC 3.9	The l2_status_lower rule runs upon receipt of the TEC_ITS_L2_NODE_STATUS event. If the l2status attribute of the received event is equal to UP, the severity of the event is set to HARMLESS.

Event clearing rules

Rule Name	TEC 3.8	TEC 3.9
interface_clearing	TEC 3.8 rule same as TEC 3.9 except hostname attribute not checked (just hostaddr, nvhostname and ifname)	The interface_clearing rule runs upon receipt of the TEC_ITS_INTERFACE_STATUS event. When this event is received, any previous status events for the same interface are downgraded to HARMLESS and closed.
isdn_clearing	TEC 3.8 rule same as TEC 3.9	The isdn_clearing rule runs upon receipt of the TEC_ITS_ISDN_STATUS event. When this event is received, any previous status events for the same ISDN interface are downgraded to HARMLESS and closed.
snmp_clearing	TEC 3.8 rule same as TEC 3.9	The snmp_clearing rule runs upon receipt of the TEC_ITS_SNMPCOLLECT_THRESHOLD event. When this event is received, any previous threshold events for the same SNMP collection daemon are downgraded to HARMLESS and closed.
node_clearing	nodestatus attribute not instantiated No check for or close of NODE_SERVICE_IMPACT events Otherwise TEC 3.8 rule same as TEC 3.9	The node_clearing rule runs upon receipt of the TEC_ITS_NODE_STATUS event. When this event is received, any previous status events for the same node are downgraded to

Rule Name	TEC 3.8	TEC 3.9
		HARMLESS and closed. If the nodestatus attribute of the received event is equal to UP, any previous TEC_ITS_NODE_SERVICE_IMPACT events for the same node are then downgraded to HARMLESS and closed.
router_clearing	routerstatus attribute not instantiated No check for or close of NODE_SERVICE_IMPACT events Otherwise TEC 3.8 rule same as TEC 3.9	The router_clearing rule runs upon receipt of the TEC_ITS_ROUTER_STATUS event. When this event is received, any previous status events for the same router are downgraded to HARMLESS and closed. If the routerstatus attribute of the received event is equal to UP, any previous TEC_ITS_NODE_SERVICE_IMPACT events for the same host are then downgraded to HARMLESS and closed.
subnet_clearing	reachability attribute not instantiated No check for or close of NODE_SERVICE_IMPACT events Otherwise TEC 3.8 rule same as TEC 3.9	The subnet_clearing rule runs upon receipt of the TEC_ITS_SUBNET_CONNECTIVITY event. When this event is received, any previous connectivity events for the same subnet are downgraded to HARMLESS and closed. If the reachability attribute of the received event is equal to REACHABLE_AGAIN, any previous TEC_ITS_SUBNET_SERVICE_IMPACT events for the same subnet are downgraded to HARMLESS and closed.
l2_status_clearing	TEC 3.8 rule same as TEC 3.9	The l2_status_clearing rule runs upon receipt of the TEC_ITS_L2_NODE_STATUS event. When this event is received, any previous L2 status events for the same node are downgraded to HARMLESS and closed.
node_service_impact_clearing	Does not exist in TEC 3.8 netview.rls	The node_service_impact_clearing rule runs upon receipt of the TEC_ITS_NODE_SERVICE_IMPACT event. When this event is received, any previous service impact events for the same node and service are downgraded to HARMLESS and closed.
subnet_service_impact_clearing	Does not exist in TEC 3.8 netview.rls	The subnet_service_impact_clearing rule runs upon receipt of the TEC_ITS_SUBNET_SERVICE_IMPACT event. When this event is received, any previous service impact events for the same subnet and service are downgraded to HARMLESS and closed.
sa_status_clearing	* saticketnumber attribute does not exist on TEC 3.8 TEC_ITS_SA_STATUS class.	The sa_status_clearing rule runs upon receipt of the TEC_ITS_SA_STATUS event. When this event is received, the event cache is searched for any previously received TEC_ITS_SA_STATUS events for the same host and with the

Rule Name	TEC 3.8	TEC 3.9
	* **sa_status_clearing_1** checks for sastatus within ifDown, nodeDown, nodeMarginal and otherwise, reception_action **clear** same in TEC 3.8 and TEC 3.9. * TEC 3.8 also has **sa_status_clearing_2** rule triggered by TEC_ITS_SA_STATUS event with sastatus within ifUp, nodeUp, nodeResolved which searches for and closes any earlier TEC_ITS_SA_STATUS events from the same node * TEC 3.8 also has **sa_status_clearing_3** rule triggered by TEC_ITS_SA_STATUS event with sastatus within ifUnmanaged, ifDeleted, nodeUnmanaged, nodeDeleted which searches for and closes any earlier TEC_ITS_SA_STATUS events from the same node	same saticketnumber attribute value. If any matching events are found, they are downgraded to HARMLESS and closed. In addition, if the sastatus attributes of the previously received event is equal to ifDown, nodeDown, or nodeMarginal, and the sastatus attribute of the new event is also equal to ifDown, nodeDown, or nodeMarginal, the severity of the new event is increased to CRITICAL.
interface_deleted	TEC 3.8 rule same as TEC 3.9	The interface_deleted rule runs upon receipt of a TEC_ITS_INTERFACE_ADDED event with action equal to DELETED. When this event is received, this rule closes any previous status events from the same interface and then **drops the received event.**
interface_unmanaged	TEC 3.8 rule same as TEC 3.9	The interface_unmanaged rule runs upon receipt of a TEC_ITS_INTERFACE_MANAGE event with manage equal to UNMANAGE. When this event is received, this rule closes any previous status events from the same interface and then **drops the received event.**
node_deleted	TEC 3.8 rule same as TEC 3.9	The node_deleted rule runs upon receipt of a TEC_ITS_NODE_ADDED event with action equal to DELETED. When this event is received, this rule closes any previous node status, router status, or interface status events from the same host and then **drops the received event.**
node_unmanaged	TEC 3.8 rule same as TEC 3.9	The node_unmanaged rule runs upon receipt of a TEC_ITS_NODE_MANAGE event with manage equal to UNMANAGE. When this event is received, this rule closes any previous node status, router status, or interface status events from the same host and then **drops the received event**.

Rule Name	TEC 3.8	TEC 3.9
interface_synchronization (change_rule)	The TEC 3.8 ruleset has a **synchronization_setup** rule at the start of this section to assert utility predicates and setup nvsync_timeout = 30 seconds, nvsync_port = 162 and nvsync_maxhosts = 10. These predicates and records have moved to the **netview_configure** rule in TEC 3.9. Otherwise TEC 3.8 rule same as TEC 3.9	The interface_synchronization change rule runs when the status of a TEC_ITS_INTERFACE_STATUS event has changed. When this occurs, the event is buffered for synchronization with the NetView component. Three types of changes are buffered: **acknowledged** status is now equal to ACK. **unacknowledged** status was previously equal to ACK, but has now been changed to something else (other than CLOSED). **closed** status has changed to CLOSED.
flush_if_ack (timer_rule)	TEC 3.8 rule same as TEC 3.9	The flush_if_ack timer rule periodically flushes acknowledged interface events and synchronizes them with the NetView component.
flush_if_unack (timer_rule)	TEC 3.8 rule same as TEC 3.9	The flush_if_unack timer rule periodically flushes unacknowledged interface events and synchronizes them with the NetView component.
flush_if_close (timer_rule)	TEC 3.8 rule same as TEC 3.9	The flush_if_close timer rule periodically flushes closed interface events and synchronizes them with the NetView component.
node_synchronization (change_rule)	TEC 3.8 rule same as TEC 3.9	The node_synchronization change rule runs when the status of a TEC_ITS_NODE_STATUS or TEC_ITS_ROUTER_STATUS event has changed. When this occurs, the event is buffered for synchronization with the NetView component. Three types of changes are buffered: **acknowledged** status is now equal to ACK. **unacknowledged** status was previously equal to ACK, but has now been changed to something else (other than CLOSED). **closed** status has changed to CLOSED.
flush_node_ack timer_rule)	TEC 3.8 rule same as TEC 3.9	The flush_node_ack timer rule periodically flushes acknowledged node or router events and synchronizes them with the NetView component.
flush_node_unack timer_rule)	TEC 3.8 rule same as TEC 3.9	The flush_node_unack timer rule periodically flushes unacknowledged node or router events and synchronizes them with the NetView component.
flush_node_close timer_rule)	TEC 3.8 rule same as TEC 3.9	The flush_node_close timer rule periodically flushes closed node or router events and synchronizes them with the NetView component.

Correlation rules

Rule Name	TEC 3.8	TEC 3.9
Service Correlation **service_impact_link_node**	Does not exist in TEC 3.8 netview.rls	The service_impact_link_node rule runs upon receipt of the TEC_ITS_NODE_SERVICE_IMPACT event. When this event is received, the rule searches the event cache for a non-closed TEC_ITS_NODE_STATUS event for the same node with nodestatus equal to DOWN, MARGINAL, or UNREACHABLE. If such a cause event is found, the two events are correlated using the link_effect_to_cause predicate, and the effect event (TEC_ITS_NODE_SERVICE_IMPACT) is acknowledged. The severity of the cause event (TEC_ITS_NODE_STATUS) is then upgraded. The new severity is FATAL if IBM Tivoli Monitoring reported a service impact; otherwise it is CRITICAL. (If the severity is already FATAL, it does not change.) If the TEC_ITS_NODE_STATUS event is linked to a non-closed NetView cause event, the new severity of the router event is propagated to its cause event.
node_link_service_impact	Does not exist in TEC 3.8 netview.rls	The node_link_service_impact rule runs upon receipt of a TEC_ITS_NODE_STATUS event with nodestatus equal to DOWN, MARGINAL, or UNREACHABLE. When this event is received, the rule searches the event cache for any non-closed TEC_ITS_NODE_SERVICE_IMPACT events for the same node. If any such effect events are found, they are correlated with the cause event using the link_effect_to_cause predicate and then acknowledged. The severity of the cause event (TEC_ITS_NODE_STATUS) is upgraded. The new severity is FATAL if IBM Tivoli Monitoring reported a service impact; otherwise, it is CRITICAL. (If the severity is already FATAL, it does not change.)

Rule Name	TEC 3.8	TEC 3.9
		If the TEC_ITS_NODE_STATUS event is linked to a non-closed NetView cause event, the new severity of the router event is propagated to its cause event.
service_impact_link_route	Does not exist in TEC 3.8 netview.rls	The service_impact_link_router rule runs upon receipt of the TEC_ITS_NODE_SERVICE_IMPACT event. When this event is received, the rule searches the event cache for a non-closed TEC_ITS_ROUTER_STATUS event for the same host with routerstatus equal to DOWN, MARGINAL, or UNREACHABLE. If such a cause event is found, the two events are correlated using the link_effect_to_cause predicate, and the effect event (TEC_ITS_NODE_SERVICE_IMPACT) is acknowledged. The severity of the cause event (TEC_ITS_ROUTER_STATUS) is then upgraded. The new severity is FATAL if IBM Tivoli Monitoring reported a service impact; otherwise it is CRITICAL. (If the severity is already FATAL, it does not change.) If the routerstatus of the TEC_ITS_ROUTER_STATUS event is equal to MARGINAL or UNREACHABLE and it is itself linked to a non-closed NetView cause event, the new severity of the router event is propagated to its cause event.
router_link_service_impact	Does not exist in TEC 3.8 netview.rls	The router_link_service_impact rule runs upon receipt of a TEC_ITS_ROUTER_STATUS event with nodestatus equal to DOWN, MARGINAL, or UNREACHABLE. When this event is received, the rule searches the event cache for any non-closed TEC_ITS_NODE_SERVICE_IMPACT events for the same host. If any such effect events are found, they are correlated with the cause event using the link_effect_to_cause predicate and acknowledged. The severity of the cause event (TEC_ITS_ROUTER_STATUS) is then upgraded. The new severity is FATAL if IBM Tivoli Monitoring reported a service impact; otherwise it is CRITICAL. (If the severity is already FATAL, it does not change.) If the routerstatus of the TEC_ITS_ROUTER_STATUS event is equal to MARGINAL or UNREACHABLE, and the event is

Rule Name	TEC 3.8	TEC 3.9
		linked to a non-closed NetView cause event, the new severity of the router event is propagated to its cause event.
service_impact_link_subnet	Does not exist in TEC 3.8 netview.rls	The service_impact_link_subnet rule runs upon receipt of the TEC_ITS_SUBNET_SERVICE_IMPACT event. When this event is received, the rule searches the event cache for a non-closed TEC_ITS_SUBNET_CONNECTIVITY event for the same subnet with reachability equal to UNREACHABLE. If such a cause event is found, the two events are correlated using the link_effect_to_cause predicate, and the effect event (TEC_ITS_NODE_SERVICE_IMPACT) is acknowledged. The severity of the cause event (TEC_ITS_SUBNET_CONNECTIVITY) is then upgraded. The new severity is FATAL if IBM Tivoli Monitoring reported a service impact; otherwise it is CRITICAL. (If the severity is already FATAL, it does not change.) If the TEC_ITS_SUBNET_CONNECTIVITY event is itself linked to a non-closed NetView cause event, the new severity of the subnet event is propagated to its cause event. If that cause event is not closed and is linked to a root cause event, the new severity is propagated to the root cause event.
subnet_link_service_impact	Does not exist in TEC 3.8 netview.rls	The subnet_link_service_impact rule runs upon receipt of a TEC_ITS_SUBNET_CONNECTIVITY event with reachability equal to UNREACHABLE. When this event is received, the rule searches the event cache for any non-closed TEC_ITS_SUBNET_SERVICE_IMPACT events for the same host. If any such effect events are found, they are correlated with the cause event using the link_effect_to_cause predicate and acknowledged. The severity of the cause event (TEC_ITS_SUBNET_CONNECTIVITY) is then upgraded. The new severity is FATAL if IBM Tivoli Monitoring reported a service impact; otherwise it is CRITICAL. (If the severity is already FATAL, it does not change.) If the TEC_ITS_SUBNET_CONNECTIVITY event is itself linked to a non-closed NetView cause event, the new severity

Rule Name	TEC 3.8	TEC 3.9
		of the subnet event is propagated to its cause event. If that cause event is not closed and is linked to a root cause event, the new severity is propagated to the root cause event.
Switch Analyzer Correlation **node_correlate_sa**	* No instantiation of severity on SA_STATUS causal event * Only action is to link effect to cause * No change of administrator, severity or status on effect NODE_STATUS event * No timer set to change status of effect event from RESPONSE to CLOSED * No propagation of severity from effect event to causal event * No action for incoming effect event of NODE_STATUS where causal SA_STATUS event has sastatus = ifDown. Instead **interface_correlate_sa** rule exists for incoming effect event of INTERFACE_STATUS where associated causal SA_STATUS event with sastatus=ifDown is imply linked to effect event	The node_correlate_sa rule runs upon receipt of a TEC_ITS_NODE_STATUS event with nodestatus equal to DOWN, MARGINAL, or UNREACHABLE. When this event is received, the rule searches the event cache for a TEC_ITS_SA_STATUS event for the same host with the sastatus attribute equal to nodeDown, nodeMarginal, or ifDown. If such a cause event is found, the following actions are taken: * The received TEC_ITS_NODE_STATUS event is linked as an effect of the TEC_ITS_SA_STATUS cause event using the link_effect_to_cause predicate. * If the sastatus attribute of the TEC_ITS_SA_STATUS cause event is equal to nodeDown or nodeMarginal, the severity of the effect event (TEC_ITS_NODE_STATUS) is set to HARMLESS, and the status of the effect event is set to RESPONSE. A timer is then set for delayed closing of the effect event after all correlation is finished. (The duration of the timer is determined by the value of the *nv_latency* global parameter.) This processing does not take place if sastatus is equal to ifDown. * The severity of the received TEC_ITS_NODE_STATUS effect event is propagated to the TEC_ITS_SA_STATUS cause event.
sa_correlate_node	* No instantiation of severity on SA_STATUS causal event * Only sastatus within nodeDown and nodeMarginal considered; sastaus=ifDown handled by separate **sa_correlate_interface** rule * Check for status on effect event NODE_STATUS only checks for outside CLOSED where TEC 3.9 checks for outside CLOSED or RESPONSE * Only action is to link effect to cause * No change of administrator, severity or status on effect NODE_STATUS event	The sa_correlate_node rule runs upon receipt of a TEC_ITS_SA_STATUS event with satatus equal to nodeDown, nodeMarginal, or ifDown. When this event is received, the rule searches the event cache for a TEC_ITS_NODE_STATUS event for the same host with nodestatus equal to DOWN, MARGINAL, or UNREACHABLE. If such an effect event is found, the following actions are taken:

Rule Name	TEC 3.8	TEC 3.9
	* No timer set to change status of effect event from RESPONSE to CLOSED * No propagation of severity from effect event to causal event	* The TEC_ITS_NODE_STATUS effect event is linked to the TEC_ITS_SA_STATUS cause event using the link_effect_to_cause predicate. * If the sastatus attribute of the TEC_ITS_SA_STATUS cause event is equal to nodeDown or nodeMarginal, the severity of the effect event (TEC_ITS_NODE_STATUS) is set to HARMLESS, and the status of the effect event is set to RESPONSE. A timer is then set for delayed closing of the effect event after all correlation is finished. (The duration of the timer is determined by the value of the *nv_latency* global parameter.) This processing does not take place if sastatus is equal to ifDown. * The severity of the received TEC_ITS_NODE_STATUS effect event is propagated to the TEC_ITS_SA_STATUS cause event.
node_up_correlate_sa	Does not exist in TEC 3.8 netview.rls	The node_up_correlate_sa rule runs upon receipt of a TEC_ITS_NODE_STATUS event with nodestatus equal to UP. When this event is received, the rule searches for any TEC_ITS_SA_STATUS events for the same host with the sastatus attribute equal to nodeUp or ifUp. If any such cause events are found, they are correlated with the effect event using the link_effect_to_cause predicate. The effect event is then downgraded to HARMLESS, its status is set to RESPONSE, and a timer is set for delayed closing of the event after all correlation is finished. (The duration of the timer is determined by the value of the *nv_latency* global parameter.)
sa_correlate_node_up	Does not exist in TEC 3.8 netview.rls	The sa_correlate_node_up rule runs upon receipt of a TEC_ITS_SA_STATUS event with satatus equal to nodeUp or ifUp. When this event is received, the rule searches the event cache for a non-closed TEC_ITS_NODE_STATUS event for the same host with nodestatus equal to UP. If such an effect event is found, it is correlated with the cause event using the link_effect_to_cause predicate. The effect event is then downgraded to HARMLESS, its status is set to RESPONSE, and a timer is set for delayed closing of the event after all correlation is finished. (The duration of the timer is determined by the value of the *nv_latency* global parameter.)

Rule Name	TEC 3.8	TEC 3.9
l2_correlate_sa	TEC 3.8 rule same as TEC 3.9	The l2_correlate_sa rule runs upon receipt of the TEC_ITS_L2_NODE_STATUS event. When this event is received, the event cache is searched for a TEC_ITS_SA_STATUS event for the same host. If such a cause event is found, the two events are correlated. In addition, one of the following actions is taken: * If the cause event has the sastatus attribute equal to ifDown, nodeDown, or nodeMarginal, it is upgraded to CRITICAL. * If the cause event has the sastatus attribute equal to ifUp, nodeUp, or nodeResolved, it is downgraded to HARMLESS. * If the cause event has the sastatus attribute equal to ifUnmanaged, ifDeleted, nodeUnmanaged, or nodeDeleted, it is upgraded to WARNING. The effect event (TEC_ITS_L2_NODE_STATUS) is then downgraded to HARMLESS and closed.
sa_correlate_l2-1	TEC 3.8 rule same as TEC 3.9	The sa_correlate_l2_1 rule runs upon receipt of a TEC_ITS_SA_STATUS event with the sastatus attribute equal to ifDown, nodeDown, or nodeMarginal. When this event is received, the event cache is searched for a TEC_ITS_L2_NODE_STATUS event for the same host. If such an effect event is found, it is correlated with the cause event using the link_effect_to_cause predicate, downgraded to HARMLESS, and closed. The cause event (TEC_ITS_SA_STATUS) is then upgraded to CRITICAL.
sa_correlate_l2-2	TEC 3.8 rule same as TEC 3.9	The sa_correlate_l2_2 rule runs upon receipt of a TEC_ITS_SA_STATUS event with the sastatus attribute equal to ifUp, nodeUp, or nodeResolved. When this event is received, the event cache is searched for a TEC_ITS_L2_NODE_STATUS event for the same host. If such an effect event is found, it is correlated with the cause event using the link_effect_to_cause predicate, downgraded to HARMLESS, and closed. The cause event (TEC_ITS_SA_STATUS) is then downgraded to HARMLESS.
sa_correlate_l2-3	TEC 3.8 rule same as TEC 3.9	The sa_correlate_l2_3 rule runs upon receipt of a TEC_ITS_SA_STATUS event with the sastatus attribute equal

Rule Name	TEC 3.8	TEC 3.9
		to ifUnmanaged, ifDeleted, nodeUnmanaged, or nodeDeleted. When this event is received, the event cache is searched for a TEC_ITS_L2_NODE_STATUS event for the same host. If such an effect event is found, it is correlated with the cause event using the link_effect_to_cause predicate, downgraded to HARMLESS, and closed. The cause event (TEC_ITS_SA_STATUS) is then set to WARNING.
router_correlate_sa	* TEC 3.8 uses first_instance to search for SA_STATUS event; TEC 3.9 uses all_instances * TEC 3.8 checks for SA_STATUS effect event with sastatus within nodeDown or nodeMarginal; TEC 3.9 also includes sastatus within ifDown * All actions otherwise similar	The router_correlate_sa rule runs upon receipt of a TEC_ITS_ROUTER_STATUS event with routerstatus equal to DOWN, MARGINAL, or UNREACHABLE. When this event is received, the event cache is searched for any TEC_ITS_SA_STATUS events for the same host with the sastatus attribute equal to nodeDown, nodeMarginal, or ifDown. If any such effect events are found, they correlated, downgraded to HARMLESS, and closed.
sa_correlate_router	* TEC 3.8 checks SA_STATUS for sastatus within nodeDown or nodeMarginal; TEC 3.9 also includes sastatus within ifDown * All actions otherwise similar	The sa_correlate_router rule runs upon receipt of a TEC_ITS_SA_STATUS event with the sastatus attribute equal to nodeDown, nodeMarginal, or ifDown. When this event is received, the event cache is searched for a TEC_ITS_ROUTER_STATUS event for the same host with routerstatus equal to DOWN, MARGINAL, or UNREACHABLE. If such a cause event is found, the two events are correlated; the effect event (TEC_ITS_SA_STATUS) is downgraded to HARMLESS and closed.
router_up_correlate_sa	Does not exist in TEC 3.8 netview.rls	The router_up_correlate_sa rule runs upon receipt of a TEC_ITS_ROUTER_STATUS event with routerstatus equal to UP. When this event is received, the event cache is searched for any TEC_ITS_SA_STATUS events for the same host with the sastatus attribute equal to nodeUp or ifUp. If any such effect events are found, they are correlated with the cause event, downgraded to HARMLESS, and closed.
sa_correlate_router_up	Does not exist in TEC 3.8 netview.rls	The sa_correlate_router rule runs upon receipt of a TEC_ITS_SA_STATUS event with the sastatus attribute equal to nodeUp or ifUp. When this event is received, the event cache is searched for a TEC_ITS_ROUTER_STATUS event for the

Rule Name	TEC 3.8	TEC 3.9
		same host with routerstatus equal to UP. If such a cause event is found, the two events are correlated; the effect event (TEC_ITS_SA_STATUS) is downgraded to HARMLESS and closed.
<u>Interface / Node Router / Subnet Correlation</u> **router_correlate_subnet**	* TEC 3.8 does not instantiate ROUTER_STATUS severity attribute * TEC 3.8 cache search for SUBNET_CONNECTIVITY events looks for status outside CLOSED, rather than CLOSED or RESPONSE * In TEC 3.8, SUBNET_CONNECTIVITY effect event has status changed to CLOSED rather than RESPONSE * No checking of status of effect event or propagation to causal ROUTER_STATUS event * No search for causal event of ROUTER_STATUS or severity propagation	The router_correlate_subnet rule runs upon receipt of a TEC_ITS_ROUTER_STATUS event with routerstatus equal to DOWN, MARGINAL, or UNREACHABLE. When this event is received, the event cache is searched for any TEC_ITS_SUBNET_CONNECTIVITY events with reachability equal to UNREACHABLE. If any such effect events are found, they are correlated using the link_effect_to_cause predicate. The effect events (TEC_ITS_SUBNET_CONNECTIVITY) are then downgraded to HARMLESS, its status is changed to RESPONSE, and a timer is set for delayed closing of the effect event after all correlation is finished. (The duration of the delay is determined by the value of the *nv_latency* global parameter.) If the severity of the effect event was higher than that of the TEC_ITS_ROUTER_STATUS cause event, the higher severity is propagated to the cause event. If routerstatus is equal to MARGINAL or UNREACHABLE and the router event is linked to a further NetView cause event, the severity is propagated to that cause event.
subnet_correlate_router	* TEC 3.8 does not instantiate SUBNET_CONNECTIVITY severity attribute * In TEC 3.8, SUBNET_CONNECTIVITY effect event has status changed to CLOSED rather than RESPONSE * No checking of status of effect event or propagation to causal ROUTER_STATUS event * No search for causal event of ROUTER_STATUS or severity propagation	The subnet_correlate_router rule runs upon receipt of a TEC_ITS_SUBNET_CONNECTIVITY event with reachability equal to UNREACHABLE. When this event is received, the event cache is searched for a TEC_ITS_ROUTER_STATUS with routerstatus equal to DOWN, MARGINAL, or UNREACHABLE. If such a cause event is found, the two events are correlated using the link_effect_to_cause predicate. The effect event (TEC_ITS_SUBNET_CONNECTIVITY) is then downgraded to HARMLESS, its status is changed to RESPONSE, and a timer is set for delayed closing of the effect event after all correlation is finished. (The duration of the delay is determined by the value of the *nv_latency* global parameter.)

Rule Name	TEC 3.8	TEC 3.9
		If the severity of the effect event was higher than that of the TEC_ITS_ROUTER_STATUS cause event, the higher severity is propagated to the cause event. If routerstatus is equal to MARGINAL or UNREACHABLE and the router event is linked to a further NetView cause event, the severity is propagated to that cause event.
interface_correlate_router	* One rule **interface_correlate_node_and_router** * TEC 3.8 does not instantiate INTERFACE_STATUS severity, date_reception and event_handle attributes (although TEC 3.9 does **not** use the date_reception or event_handle variables) * In TEC 3.8 the **interface** event is ALWAYS the causal event; in TEC 3.9 the **interface** event is causal only if the router status is MARGINAL or UNREACHABLE * Effect event is CLOSED in TEC 3.8, rather than set to RESPONSE * No severity propagation from effect to causal event	The interface_correlate_router rule runs upon receipt of a TEC_ITS_INTERFACE_STATUS event with ifstatus equal to DOWN, ADMIN_DOWN, or UNREACHABLE. When this event is received, the event cache is searched for a TEC_ITS_ROUTER_STATUS event from the same host. If such an event is found, one of the following actions is taken: * If the TEC_ITS_ROUTER_STATUS event has routerstatus equal to MARGINAL or UNREACHABLE, it is correlated as an effect event using the link_effect_to_cause predicate, it is downgraded to HARMLESS, and its status is set to RESPONSE. A timer is then set for delayed closing of the effect event after all correlation is finished. (The duration of the delay is determined by the value of the *nv_latency* global parameter.) If the severity of the effect event was higher than that of the TEC_ITS_INTERFACE_STATUS cause event, the higher severity is propagated to the cause event. * If the TEC_ITS_ROUTER_STATUS event has routerstatus equal to DOWN, it is correlated as a cause event using the link_effect_to_cause predicate. The effect event (TEC_ITS_INTERFACE_STATUS) is then downgraded to HARMLESS and closed.
router_correlate_interface	* TEC 3.8 does not instantiate ROUTER_STATUS severity, date_reception and event_handle attributes (although TEC 3.9 does **not** use the date_reception or event_handle variables) * In TEC 3.8 the **interface** event is ALWAYS the causal event; in TEC 3.9 the **interface** event is causal only if the router status is MARGINAL or UNREACHABLE * Effect event is CLOSED in TEC 3.8, rather than set to RESPONSE	The router_correlate_interface rule runs upon receipt of a TEC_ITS_ROUTER_STATUS event with routerstatus equal to DOWN, MARGINAL, or UNREACHABLE. When this event is received, one of the following actions is taken: * If the routerstatus is equal to DOWN, the event cache is searched for any TEC_ITS_INTERFACE_STATUS events with ifstatus equal to DOWN, ADMIN_DOWN, or UNREACHABLE. If any are found, they are correlated as

Rule Name	TEC 3.8	TEC 3.9
	* No severity propagation from effect to causal event	effect events using the link_effect_to_cause predicate. These effect events are then downgraded to HARMLESS and closed. * If the routerstatus is not equal to DOWN, the event cache is searched for a TEC_ITS_INTERFACE_STATUS event with ifstatus equal to DOWN, ADMIN_DOWN, or UNREACHABLE. If such an event is found, it is correlated as the cause event using the link_effect_to_cause predicate. The effect event (TEC_ITS_ROUTER_STATUS) is then downgraded to HARMLESS, its status is changed to RESPONSE, and a timer is set for delayed closing of the effect event after all correlation is finished. (The duration of the delay is determined by the value of the *nv_latency* global parameter.) If the severity of the effect event was higher than that of the TEC_ITS_INTERFACE_STATUS cause event, the higher severity is propagated to the cause event.
interface_correlate_node	* In TEC 3.8 the **interface** event is the causal event; in TEC 3.9 the **node** event is causal	The interface_correlate_node rule runs upon receipt of a TEC_ITS_INTERFACE_STATUS event with ifstatus equal to DOWN, ADMIN_DOWN, or UNREACHABLE. When this event is received, the event cache is searched for **a** TEC_ITS_NODE_STATUS event for the same host with nodestatus equal to DOWN, MARGINAL, or UNREACHABLE. If such a cause event is found, the two events are correlated using the link_effect_to_cause predicate. The effect event (TEC_ITS_INTERFACE_STATUS) is then downgraded to HARMLESS and closed.
node_correlate_interface	* In TEC 3.8 the **interface** event is the causal event; in TEC 3.9 the **node** event is causal * TEC 3.8 uses first_instance to search event cache; TEC 3.9 uses all_instances to search for interface effect events	The node_correlate_interface rule runs upon receipt of a TEC_ITS_NODE_STATUS event with nodestatus equal to DOWN, MARGINAL, or UNREACHABLE. When this event is received, the event cache is searched for **any** TEC_ITS_INTERFACE_STATUS events for the same host with ifstatus equal to DOWN, ADMIN_DOWN, or UNREACHABLE. If any such effect events are found, they are

Rule Name	TEC 3.8	TEC 3.9
		correlated using the link_effect_to_cause predicate, downgraded to HARMLESS, and closed.
router_up_correlate_subnet	Does not exist in TEC 3.8 netview.rls	The router_up_correlate_subnet rule runs upon receipt of a TEC_ITS_ROUTER_STATUS event with routerstatus equal to UP. When this event is received, the event cache is searched for any TEC_ITS_SUBNET_CONNECTIVITY events with reachability equal to REACHABLE_AGAIN. If any such effect events are found, they are correlated using the link_effect_to_cause predicate. The effect events (TEC_ITS_SUBNET_CONNECTIVITY) are then downgraded to HARMLESS and closed.
subnet_correlate_router_up	Does not exist in TEC 3.8 netview.rls	The subnet_correlate_router_up rule runs upon receipt of a TEC_ITS_SUBNET_CONNECTIVITY event with reachability equal to REACHABLE_AGAIN. When this event is received, the event cache is searched for a TEC_ITS_ROUTER_STATUS event with routerstatus equal to UP. If such a cause event is found, the two events are correlated using the link_effect_to_cause predicate. The effect event (TEC_ITS_SUBNET_CONNECTIVITY) is then downgraded to HARMLESS and closed.
interface_up_correlate_router_up	Does not exist in TEC 3.8 netview.rls	The interface_up_correlate_router_up rule runs upon receipt of a TEC_ITS_INTERFACE_STATUS event with ifstatus equal to UP. When this event is received, the event cache is searched for **a** TEC_ITS_ROUTER_STATUS event for the same host with routerstatus equal to UP. If such a cause event is found, the two events are correlated using the link_effect_to_cause predicate. The effect event is then downgraded to HARMLESS and closed.
interface_up_correlate_node_up	Does not exist in TEC 3.8 netview.rls	The interface_up_correlate_node_up rule runs upon receipt of a TEC_ITS_INTERFACE_STATUS event with ifstatus equal to UP. When this event is received, the event cache is searched for **a** TEC_ITS_NODE_STATUS event for the same host with nodestatus equal to UP. If such a cause event is found, the two events are correlated using the link_effect_to_cause predicate. The effect event (TEC_ITS_INTERFACE_STATUS) is then downgraded to HARMLESS and closed.

Rule Name	TEC 3.8	TEC 3.9
node_up_correlate_interface_up	Does not exist in TEC 3.8 netview.rls	The node_up_correlate_interface_up rule runs upon receipt of a TEC_ITS_NODE_STATUS event with nodestatus equal to UP. When this event is received, the event cache is searched for **any** TEC_ITS_INTERFACE_STATUS events for the same host with ifstatus equal to UP. If any such effect events are found, they are correlated using the link_effect_to_cause predicate, downgraded to HARMLESS, and closed.
router_up_correlate_interface_up	Does not exist in TEC 3.8 netview.rls	The router_up_correlate_interface_up rule runs upon receipt of a TEC_ITS_ROUTER_STATUS event with routerstatus equal to UP. When this event is received, the event cache is searched for any TEC_ITS_INTERFACE_STATUS events for the same host with ifstatus equal to UP. If any such effect events are found, they are correlated using the link_effect_to_cause predicate, downgraded to HARMLESS, and closed.
ITM Correlation **subnet_correlate_unreachable**	Same in TEC 3.8 netview.rls except no search for TEC_Heartbeat_missed events	The subnet_correlate_unreachable rule runs upon receipt of a TEC_ITS_SUBNET_CONNECTIVITY event. When this event is received, the following actions are taken: * If the reachability attribute of the received event is equal to UNREACHABLE, a subnet-unreachable fact is asserted in the knowledge base. If reachability is equal to REACHABLE_AGAIN, the subnet-unreachable fact is retracted. * If the reachability attribute of the received event is equal to UNREACHABLE, the event cache is searched for any TEC_ITS_UNREACHABLE events from the same subnet. If any such events are found, they are correlated as effect events, downgraded to HARMLESS, and closed. * If the reachability attribute of the received event is equal to UNREACHABLE, the event cache is searched for any TEC_Heartbeat_missed events from the same subnet. If any such events are found, they are correlated as effect events, downgraded to HARMLESS, and closed.
unreachable_correlate_subnet	Same in TEC 3.8 netview.rls	The unreachable_correlate_subnet rule runs upon receipt of a TEC_ITS_UNREACHABLE event. When this event is received, the knowledge base is checked for any subnet-unreachable facts related to the subnet of the IP address

Rule Name	TEC 3.8	TEC 3.9
		that is unreachable. If a subnet-unreachable fact is found, the event cache is then searched for a TEC_ITS_SUBNET_CONNECTIVITY event related to the specified subnet. If this cause event is found, it is correlated with the TEC_ITS_UNREACHABLE effect event using the link_effect_to_cause predicate. The effect event is then downgraded to HARMLESS and closed.
heartbeat_missed_link_subnet	Does not exist in TEC 3.8 netview.rls	The heartbeat_missed_link_subnet rule runs upon receipt of a TEC_Heartbeat_missed event. When this event is received, the knowledge base is checked for any subnet-unreachable facts related to the subnet of the IP address that sent the event. If a subnet-unreachable fact is found, the event cache is then searched for a TEC_ITS_SUBNET_CONNECTIVITY event related to the specified subnet. If this cause event is found, it is associated with the TEC_Heartbeat_missed effect event using the link_effect_to_cause predicate.
node_link_hearbeat_missed	Does not exist in TEC 3.8 netview.rls	The node_link_heartbeat_missed rule runs upon receipt of a TEC_ITS_NODE_STATUS event with nodestatus equal to any value other than UP. When this event is received, the event cache is searched for any TEC_Heartbeat_missed events for the same host. (If the TEC_ITS_NODE_STATUS event does not have a value for the **fqhostname** attribute, the hostname attribute is compared to the fqhostname attribute of the heartbeat event. This comparison is not case sensitive.) If any such effect events are found, they are associated using the link_effect_to_cause predicate.
heartbeat_missed_link_node	Does not exist in TEC 3.8 netview.rls	The heartbeat_missed_link_node rule runs upon receipt of a TEC_Heartbeat_missed event. When this event is received, the event cache is searched for a TEC_ITS_NODE_STATUS event for the same host with nodestatus equal to anything other than UP. (If the TEC_ITS_NODE_STATUS event does not have a value for the fqhostname attribute, the hostname attribute is compared to the **fqhostname** attribute of the heartbeat event. This comparison is not case sensitive.) If such a cause event is found, the two events are associated using the link_effect_to_cause predicate.

Rule Name	**TEC 3.8**	**TEC 3.9**
router_link_hearbeat_missed	Does not exist in TEC 3.8 netview.rls	The router_link_heartbeat_missed rule runs upon receipt of a TEC_ITS_ROUTER_STATUS event with routerstatus equal to any value other than UP. When this event is received, the event cache is searched for any TEC_Heartbeat_missed events for the same host. (If the TEC_ITS_ROUTER_STATUS event does not have a value for the **fqhostname** attribute, the hostname attribute is compared to the fqhostname attribute of the heartbeat event. This comparison is not case sensitive.) If any such effect events are found, they are associated using the link_effect_to_cause predicate.
heartbeat_missed_link_router	Does not exist in TEC 3.8 netview.rls	The heartbeat_missed_link_router rule runs upon receipt of a TEC_Heartbeat_missed event. When this event is received, the event cache is searched for a TEC_ITS_ROUTER_STATUS event for the same host with routerstatus equal to anything other than UP. (If the TEC_ITS_ROUTER_STATUS event does not have a value for the **fqhostname** attribute, the hostname attribute is compared to the fqhostname attribute of the heartbeat event. This comparison is not case sensitive.) If such a cause event is found, the two events are associated using the link_effect_to_cause predicate.
delayed_close **(timer_rule)**	Does not exist in TEC 3.8 netview.rls	The delayed_close timer rule downgrades to HARMLESS and closes any event that was scheduled for delayed closing pending the completion of correlation.