Cisco Configuration Assurance Solution – SPM Audit and Analysis SP Sentinel Reference Guide

Software Release 11.5

Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

Text Part Number: OL-8384-01

Cisco Configuration Assurance Solution – SPM
Audit and Analysis
SP Sentinel Reference Guide

Copyright

Document Copyright

Document Title: Reference Guide
Document Part Number: D00266
Version: 6

Software Copyright

Product Name: SP Sentinel
Product Release: 11.5

Documentation Conventions

OPNET documentation uses specific formatting and typographic conventions to present the following types of information:

- Objects, examples, and system I/O
- Object hierarchies, notes, and warnings
- Computer commands
- Lists and procedures

Objects, Examples, and System I/O

- Directory paths and file names are in plain Courier typeface:

  `opnet\release\models\std\ip`

- Function names in body text are in italics:

  op_dist_outcome()

- The names of functions of interest in example code are in bolded Courier typeface:

```
/* determine the object ID of packet's creation module */
src_mod_objid = op_pk_creation_mod_get (pkptr);
```

- Variables are enclosed in angle brackets (< >):

  `<opnet_user_home>/op_admin/err_log`

Object Hierarchies, Notes, and Warnings

Menu hierarchies are indicated by right angle brackets (>); for example:

Open File > Print Setup > Properties...

Attribute hierarchies are represented by angled arrows (↘) that indicate that you must drill down to a lower level of the hierarchy:

Attribute level 1 ➘ Attribute level 2 ➘ Attribute level 3

Note—Notes are indicated by text with the word Note at the beginning of the paragraph. Notes advise you of important supplementary information.

WARNING—Warnings are indicated by text with the word WARNING at the beginning of the paragraph. Warnings advise you of vital information about an operation or system behavior.

Computer Commands

These conventions apply to windowing systems and navigation methods that use the standard graphical-user-interface (GUI) terminology such as click, drag, and dialog box.

- Key combinations appear in the form "press **<button>+x**"; this means press the **<button>** and **x** keys *at the same time* to do the operation.
- The mouse operations *left-click* (or *click*) and *right-click* indicate that you should press the left mouse button or right mouse button, respectively.

Lists and Procedures

Information is often itemized in bulleted (unordered) or numbered (ordered) lists:

- In bulleted lists, the sequence of items is not important.
- In numbered lists, the sequence of items is important.

Procedures are contained within procedure headings and footings that indicate the start and end of the procedure. Each step of a procedure is numbered to indicate the sequence in which you should do the steps. A step may be followed by a description of the results of that step; such descriptions are preceded by an arrow.

Procedure FM-1 Sample Procedure Format

1 Procedure step.

➥ Result of the procedure step.

2 Procedure step.

End of Procedure FM-1

For more information about using and maintaining OPNET documentation, see the Documentation Guide.

Document Revision History

Release Date	Product Version	Chapter	Description of Change
August 2005	11.5	Preferences	• Added Standard preference handle_exception_extended. • Added GUI preferences allow_rotated_world_coordinates, flow_spreadsheet_import_filename, flow_spreadsheet_import_overwrite, network_palette.style, show_network_browser_threshold, subnet_hierarchy.interactive.reparenting_disable, suppress_use_startup_wizard_checkbox, and use_startup_wizard.
February 2005	11.0 PL3	Preferences	• Clarified precedence of Assignment Mechanisms.
		Program Descriptions	• 64-bit kernels now supported on Windows.
October 2004	11.0 PL1	Preferences	• Added new GUI preferences icon_autosizing.disable, icon_autosizing.large_element_count, icon_autosizing.max_neighbors, network_layout.adjust_unconnected_groups, network_layout.max_overlap_links_to_bend, network_layout.use_simple_threshold, title_autoplacing.directions, title_autoplacing.disable, and title_autoplacing.try_small_font.
		Program Descriptions	• Expanded description of parallel_sim.num_processors preference.
August 2004	11.0	Preferences	• Added new preferences geocentric_model and code_editor_prog. • Removed /G6 from the comp_flags_optim default. • Added section on Passwords.
		Program Descriptions	• Deleted obsolete preference parallel_sim.ui_selected.
March 2004	10.5 PL1	Distributions	• Added chapter describing distributions
		PDF Editor	• Added chapter describing PDF Editor
January 2004	10.5	Protocol Models	• Deleted this chapter from the manual. For protocol information, see the Standard Model User Guides and Specialized Model User Guides in this document set.
		System Environment	• Added information on File Names for Compiled Code. • Replaced .sid and .sio with .repos in File Type Suffixes. • Added Windows file extensions for object files and libraries in File Type Suffixes.
		Environment Attributes	• Moved details on Architecture Suffix to *System Environment* chapter.

Release Date	Product Version	Chapter	Description of Change
September 2003	10.0	Revision History	Section added to this manual.

Contents

List of Figures

List of Tables

List of Procedures

1 Protocol Models

This document describes the features included in the following protocol models:

ATM

Asynchronous Transfer Mode (ATM) is a connection-oriented packet switching technique that is universally accepted as the transfer mode of choice for Broadband Integrated Services Digital Network. This document describes key features of the ATM model suite shipped as part of the standard OPNET model library.

Model Features

This section provides a list of the main features available in the ATM model:

- The ATM model suite captures the following protocol behavior:

Table 1-1 ATM Model Features

Feature	Description
Signaling Support	Signaling is provided for point-to-point, full-duplex, Switched Virtual Circuit (SVC), Soft-Permanent Virtual Circuit (SPVC) and Soft-Permanent Virtual Path (SPVP). Dynamic call setup and teardown procedures are supported.
Traffic Control	Traffic control includes Call Admission Control (CAC) and Usage Parameter Control (UPC). Traffic Control is based on specific service category, traffic parameters (PCR, SCR, MCR, MBS) and QoS parameters (ppCDV, maxCTD, CLR). Traffic Control prevents all calls with unsupportable traffic requirements from establishing a connection and prevents established calls from degrading network service below the specified Quality of Service requirements.
Buffering	Output port buffering is modeled. Output buffer supports the round-robin and weighted round-robin queueing schemes. Buffers can be configured at each switch for various QoS levels. A QoS level is made up of the QoS category (CBR, rt-VBR, nrt-VBR, ABR, UBR), the QoS parameters, (ppCDV, max CTD, CLR), and the traffic parameters.
Connection support	The connections of the ATM model suite enable you to do the following: • Configure hard and soft PVPs and PVCs • Switch between hard and soft PVPs and PVCs • Import hard and soft PVCs from a file • Specify a preferred route for an SPVx • Accelerate capacity planning using the Call Only Mode feature on ATM uni-source nodes • Set up SVC connections on a per-demand basis • Display VC routes using the route browser • ATM connection models are implemented based on information available from the sources listed below and in Table 1-2.
End of Table 1-1	

- Configurable parameters for Service Specific Connection Oriented Protocol (SSCOP)

- Dynamic routing using PNNI or an adaptation of the Bellman-Ford shortest path algorithm
- Simulation efficiency modes to reduce the duration of large simulations without sacrificing accuracy
- An API package that applications can use to create, destroy, and send packets through an ATM Virtual Circuit
- SPVC reroute capability to reroute SPVC connections on failure
- ATM models are implemented based on information available from the following sources

Table 1-2 Reference Documents

ATM Forum, September 1993	ATM User-Network Interface Specification, Version 3.0
ATM Forum, April 1996	ATM Traffic Management Specification, Version 4.0
ITU-TSS Recommendation I.150	B-ISDN ATM Functional Characteristics
ITU-TSS Recommendation I.361	B-ISDN ATM Layer Specification
ITU-TSS Recommendation I.362	B-ISDN ATM Adaptation Layer (AAL) Functional Description
ITU-TSS Recommendation I.363	B-ISDN ATM Adaptation Layer (AAL) Specification
ITU-T Specification Q.2110	B-ISDN AAL - Service Specific Connection Oriented Protocol (SSCOP)
P-NNI V1.0	PNNI 1.0 ATM Forums: Private Network-Network Interface Specification Version 1.0 (PNNI1.0)
End of Table 1-2	

BGP

Border Gateway Protocol (BGP) is an interdomain routing protocol used to exchange network reachability information between BGP speaking systems. Unlike most other routing protocols that operate within an autonomous system (such as RIP and OSPF), BGP is designed to transmit routing information between different autonomous systems. The OPNET BGP model implements the current version: BGP-4. This document describes the BGP model distributed with the Standard Model Library.

Model Features

The BGP model supports the following protocol features.

Table 1-3 BGP Model Features (Part 1 of 2)

Feature	Description
Neighbor configuration	IBGP (Internal BGP) and EBGP (External BGP) neighbors can be configured for a BGP router.
Routing policies	A router can be configured with multiple routing policies. These routing policies can be applied to neighbors to restrict outgoing or incoming routes or to modify the attributes of advertised routes. The following attributes can be checked or modified in route maps: • IP Address • AS Path • Communities • Local Preference • Degree of Preference • Multi Exit Discriminator • Weight
Route filtering	Route filtering, on prefix lists and AS path lists, is supported on a per-router or per-neighbor basis.
Redistribution	Routes can be redistributed between BGP and other routing protocols. You can use route maps to restrict or change the attributes of routes redistributed into BGP according to the following: • Next hop • Metric • Route type (Internal or External) • Destination’s IP address

Table 1-3 BGP Model Features (Part 2 of 2)

Feature	Description
Advertised destinations	A BGP router can be configured to advertise additional routes. That is, the router can add routes that may not be included in the routes received from neighbors and other routing protocols.
Confederations	Large autonomous systems may be split into smaller member autonomous systems to reduce the required number of IBGP peering sessions.
Route Reflectors	The use of route reflectors to reduce the IBGP mesh is supported.
Timers	The following timers are modeled. • Hold Timer. The value of this timer is the maximum interval during which the routes advertised by a peer will be valid without a keepalive or an update packet coming from the peer. The hold timer is started once the connection to the peer is established and it will be periodically reset every time a keepalive or an update packet is received from the peer. • Keep Alive Timer. This timer value is not taken from an attribute, but is modeled internally as one third of the value of the Hold Timer. • Connect Retry Timer. Time between successive attempts to open a TCP connection with a neighbor. This timer is canceled once the TCP connection is successfully established. • Min Route Advertisement Interval. Minimum amount of time that must elapse between advertisements of routes to a particular destination from a single BGP speaker. • Scan Period. The period between two successive BGP scans. A BGP scan determines the metric changes to routes specified in the Network Reachability Attribute and advertises them via UPDATE messages. It also detects if previously unreachable neighbors have become reachable and triggers the connection establishment mechanism.
End of Table 1-3	

Reference Documents

The model is implemented based on information from the following sources.

Table 1-4 Reference Documents

RFC-1771-1774	BGP4 Specifications and usage guidelines
RFC-1997, 1998	BGP communities usage
RFC-2796	Use of route reflectors
RFC-3065	BGP Confederations specifications
End of Table 1-4	

EIGRP

EIGRP (Enhanced Interior Gateway Routing Protocol) is a distance-vector routing protocol developed by Cisco Systems to reduce routing protocol overhead in steady state, and to provide fast convergence in case of topology changes. In well-designed networks, EIGRP scales well and converges quickly with minimal network traffic. To minimized its load on the network, EIGRP propagates only routing table changes instead of the entire routing table when a change occurs.

This document describes the EIGRP model, which is shipped as part of the standard OPNET model library.

Model Features

The following table summarizes the main EIGRP features included in the model implementation.

Table 1-5 EIGRP Model Features

Feature	Description
Neighbor Discovery/Recovery	The model implements automatic neighbor discovery and recovery at start time and after failures. Hello, update, query, and reply messages are explicitly modeled.
Timers	Hold Time and Hello Interval timers are configurable on a per-AS basis on each interface basis.
EIGRP Tables	The following tables are modeled: • Topology table • Routing table • Neighbor table
DUAL	The model handles topology changes—such as node and link failures and recoveries—using the Diffusing Update Algorithm (DUAL) when necessary.
Metric Computations	The model uses composite metric computation using configurable K values. Metric variance is also configurable.
Load Balancing	Load balancing over multiple routes is modeled.
Passive Interfaces	The model supports the passive interface feature for EIGRP. Passive interfaces do not send or receive routing updates; however, they are included in routing updates sent from non-passive interfaces.
Split-Horizon	Split horizon can be enabled on a per-interface basis to prevent routing loops.
Route redistribution	The EIGRP model accepts redistributed external routes.
Route Filtering	Incoming and outgoing routes can be filtered according to prefix lists on a per-interface basis.
Multiple Processes	Multiple EIGRP processes can be active on a router.
VLSM	Variable length subnet masks are supported.
Traffic Sharing	Traffic sharing is configurable on a router-wide or per-interface basis.
End of Table 1-5	

Model Limitations

The following protocol features are not implemented in the EIGRP model suite.

- The model suite does not include the implementation of the RTP (Reliable Transfer Protocol) component of EIGRP. The model assumes all the underlying protocol stacks are reliable and every packet sent reaches its destination unless there is a link or node failure (that is, no underlying protocol packet drops).
- Among the protocol dependent modules, only the IP module is included in the implementation—the EIGRP module does not run over the IPX module.
- Stub routers and multipath routes threshold are not supported.

Reference Documents

EIGRP is a proprietary product of Cisco Systems, so there is no standard that explains the design and details of the protocol. The EIGRP model is based on CISCO product documentation and on the following references.

Table 1-6 Reference Documents

Jeff Doyle, CCIE#1919	CCIE Professional Development: Routing TCP/IP, Volume 1
Alvaro Ratana, Russ White, Don Slice	EIGRP for IP: Basic Operation and Configuration, Addison Wesley Networking Basics Series, 2000.
End of Table 1-6	

Protocol Overview

EIGRP activities start at EIGRP start time by sending initial hello messages through the active interfaces of the underlying router node. The EIGRP node discovers neighbor EIGRP modules when they send hello messages, and the module begins to build its topology and routing tables accordingly. In an effort to discover the entire network, the EIGRP nodes then exchange topology and update messages with their new-found neighbors. When all update messages have been processed and there are no triggered update messages left, the EIGRP network has converged (that is, all routers have discovered all of the IP subnetworks in the entire network). At this point, the EIGRP process enters into steady state.

In steady state, periodic hello messages are used by EIGRP nodes to detect changes in neighbor node status. If an EIGRP node does not receive a hello message from a neighbor for a specific amount of time, the EIGRP process considers that neighbor to be down or unreachable. If the failed node recovers, then its EIGRP process will start sending periodic hello messages again, and these hello messages alert its neighbors that it has recovered.

In case of topology changes, the EIGRP process tries to recover quickly by using its topology table to find an alternate route. If the topology table does not yield a feasible alternate route for some subnetworks, the EIGRP process starts DUAL, which queries neighbor nodes' topology tables for an alternate route. Execution of DUAL causes query and reply message exchanges between EIGRP processes, as well as the exchange of update messages. The EIGRP network reaches convergence again when there are no active query, reply or update messages in the network. Any subnetworks to which DUAL could not find a route become unreachable.

Ethernet

Ethernet is a bus-based local area network (LAN) technology commonly used in the technical and business communities.

Detailed information about the Ethernet protocol is in the *IEEE 802.3 Standard.*

Model Features

The Ethernet MAC model provided with OPNET implements the carrier sensing, collision detection, and retransmission mechanisms specified in the *IEEE 802.3, IEEE 8-2.3u, and IEEE 802.3z Standard*. Explicit modeling is performed for all features other than serialization of bit transfers to and from the physical layer. The following list itemizes the features provided in this model:

- FIFO processing of Transmission Requests
- Propagation Delay based on Distance Between Individual Stations
- Carrier Sensing from Physical Layer
- Collision Detection from Physical Layer
- Truncated Binary Exponential Backoff
- Transmission Attempt Limit of 16
- Interframe Gap Timing for Deference
- Jam Sequence Transmission after Collisions
- 802.3 Minimum and Maximum Frame Sizes
- Frame Bursting (1000BaseX Ethernet operating in half-duplex only)
- Full- and half-duplex transmissions

You can configure port-based VLANs on all generic bridge and switch models, and on any vendor-specific models that support this technology. Ethernet link models allow you to simulate point-to-point trunk links; a single trunk link can carry traffic for multiple VLANs as specified by IEEE 802.1q.

To configure a VLAN, set the VLAN Scheme attribute to “Port-based VLAN” on the bridge or switch supporting the VLAN. You can assign VLAN identifiers to specific port numbers in the VLAN Port Configuration Table. (To find a link’s port numbers, use Link Interfaces on the Edit Attributes (Advanced) dialog box.) Note that you can assign only one VLAN identifier to a specific port. However, multiple ports can belong to the same VLAN.

The Ethernet models also support Fast EtherChannel technology. This allows multiple Ethernet point-to-point links to be bundled into one logical full-duplex channel of up to 800 Mbps (for Fast Ethernet) or 8000 Mbps (for Gigabit Ethernet). You can use a Fast EtherChannel or Gigabit EtherChannel link in place of any regular Ethernet link (10BaseT, 100BaseT, or 1000BaseX). EtherChannel links support flow-based balancing of traffic, and are useful for upgrading bottleneck links in Ethernet LAN networks.

Note—You can only use EtherChannel links when Ethernet is running in full-duplex mode.

The Ethernet models can be deployed either in a bus (10Base2) or a hub (10BaseT, 100BaseT or fast ethernet, and 1000BaseX or gigabit ethernet) configuration. The following list itemizes the main differences between these two configurations:

- Connections from the MAC processes to the hub are via duplex point-to-point links, as opposed to a bus medium.
- Collision Detection in the hub configuration is handled by the hub, rather than individual MAC processes.
- Deference mechanism is handled by the hub, rather than a separate deference process.
- Ethernet hubs cannot be directly connected to one another. Instead, a bridge must be used to link two or more hubs together.

Frame Relay

Frame Relay is a connection-oriented, packet-switching protocol used to pass information across a digital interface. “Connection-oriented” means that a connection between the communicating parties is set up before the actual data transfer takes place. Other higher layer protocols are required to handle additional network services, such as flow control and error recovery. This protocol operates in the lowest sublayer of the data link layer (DLL) of the Open System Interconnection (OSI) reference model.

Detailed information on the Frame Relay protocol can be found in the following documentation:

- ANSI T1.606, 1990, Frame Relay Bearer Service.
- ANSI T1.617, 1991, Signaling Specification for Frame Relay Bearer Service.
- ANSI T1.618, 1991, Core Aspects of Frame Protocol for Use with Frame Relay Bearer Service.
- ANSI T1.606a, 1992, Frame Relay Bearer Service - Congestion Management and Frame Size.
- ANSI T1.513, 1994, User Information Transfer Performance Parameters.

These documents can be obtained from the American National Standards Institute.

Additionally, RFC-1490 has been used as a reference for the multiprotocol interconnection over Frame Relay.

The following is a list of additional ANSI standards available on the Frame Relay protocol:

- ANSI T1.606b, 1993, Frame Relay Bearer Service - Network-to-Network Interface Requirements.
- ANSI T1.617a, 1994, Signaling Specification for Frame Relay Bearer Service - Protocol Encapsulation and PICS.

Model Features

This section discusses the implementation choices made in developing this OPNET Frame Relay model. Certain parts of the protocol have been simplified or omitted in view of the fact that it is intended primarily for performance estimation.

The following information is important in order to determine whether the OPNET Frame Relay example model is applicable for a particular simulation study or not.

The main limitations of the described model are:

- The model does not incorporate the SVC implementation for call establishment; rather all the virtual circuits used for data transfer are modeled using the PVC approach. The PVC approach assumes that the user sets up the virtual circuits at the service subscription time and these virtual circuits remain intact during the length of the simulation. Specific instructions for setting up a PVC connection between source-destination application pairs are provided later in this document.
- The *Local Management Interface* (LMI) message passing is not implemented. The LMI messages, if implemented, help in providing the user with status and configuration information for the PVCs operating over the Frame Relay interface. This feature is not implemented based on the assumption that once the PVC circuits are setup, they are not released until the end of the simulation.
- The *Consolidated Link Layer Management* (CLLM) message provides a useful feature to the Frame Relay protocol by acting as an additional congestion notification mechanism. However, since this feature is optional and many systems do not implement this feature, the current model does not support it.
- The frame check sequence (FCS) checking is not implemented. This feature of the Frame Relay protocol is used to ensure that transmission errors get trapped by the receiving stations. However, assuming that transmission errors are very rare, the implementation of this feature has been omitted from the simulation model.
- The model provides only a *store-and-forward* switching methodology (i.e., the frame switching takes place within a network switching device only after it is completely received). There may be instances when the frame switching equipment employs a *cut-through* switching technology (i.e., the leading edge of the incoming frame may be forwarded on to the next link before the trailing edge is completely received). If *cut-through* switching needs to be modeled, changes will have to be made to the existing model to achieve the desired characteristics.

Some of the important features that the model incorporates are:

- The Frame Relay example model is developed in a modular fashion, with simple interface requirements for DTE and DCE. The FRAD implementation takes care of interfacing the application to the Frame Relay network. Applications may be supported directly on top of the FRAD, or via an interface module if the applications relies on some other protocol. As an example, a Frame Relay-IP interface model (FRIPIF) is provided to illustrate interfacing Frame Relay with IP (an example showing how to set up applications using TCP/IP to support data transfer over a Frame Relay network is also provided).

- Each PVC supports separate bi-directional specification of all the necessary parameters associated with it (i.e., CIR, Bc and Be along with the source and destination application information). More details on this topic are provided in a later section of this document.

- The model implements congestion control mechanisms at the DTE on a per-PVC basis by monitoring the data transfer in accordance with the subscription time parameters (CIR, B_c and B_e) for that PVC over a measurement interval, T_c. Data in excess of the B_c specification is marked DE, and data in excess of B_e specification is discarded immediately.

- The model allows emulation of zero CIR specification. In this mode all the frames generated at the source are marked *Discard Eligible* (DE).

- The model implements congestion notification at the DCE by monitoring network resource utilization in terms of the buffer usage for the outgoing link supporting each PVC. If the buffer usage exceeds a certain preset threshold (to denote onset of congestion), the FECN and/or BECN bits are marked in the outgoing frames. The model allows the user to define the following thresholds (expressed as a percentage of the total bit capacity for that buffer) where congestion control mechanisms are affected:

  — Enable Congestion Status

  — Disable network congestion

  The model also implements congestion control at the DCE by discarding frames which are discard eligible (i.e., with their DE bit set), and in case of severe throttling discarding all the frames until the network recovers from the congestion status. Again, these mechanisms are activated when the buffer usage exceeds predefined thresholds. The following four attributes are used to set these thresholds:

  — Start discarding DE frames

  — Start discarding all frames

  — Stop discarding all frames

  — Stop discarding DE frames

Additionally, the model provides an option for the network to participate in congestion control by setting the DE bit in frames if the network is in a congested state. This feature can be incorporated in the model by enabling the following parameter:

— Network action on DE bit

All of the seven parameters above are attributes of the Frame Relay switching node. The values for these parameters are computed as a percentage of the buffer capacity for that switching node.

- The model implements Frame Relay fragmentation. The Frame Relay layer fragments application packets that are larger than a specified maximum frame size that the network can carry. At the destination end, these fragments are reassembled to form the original application data packet.
- The model calculates standard user-information-transfer performance parameters like end-to-end delays, congestion status of the network with respect to time, and the frame residual error rate (RER). More details on these statistics can be found in a subsequent section of this document. Additionally, specific portions of the model can be easily modified to incorporate other custom statistics, if desired.
- The number of application processes running directly on top of the FRAD can be easily changed without modifying any of the existing models. More details on this topic are provided in a later section of this document.
- The Frame Relay demand object provides you with a way to define Frame Relay PVCs. You define the PVC as a demand object overlaid between two nodes in a network model, then you can define specific characteristics of the PVC on the Layer 2 Mapping compound attribute of the interface. You do not need to use the Frame Relay PVC Config node in the model.
- The Frame Relay PVC Config node allows you to define a traffic pattern and assign it to a PVC. You can also create multiple PVCs for a given source-destination pair; this allows you to characterize different PVCs to support specific traffic patterns, thereby providing varying QoS for different higher-layer traffic flows.
- The standard model library includes a set of Frame Relay “cloud” models that allow you to abstract parts of a network infrastructure. A cloud behaves like a simplified Frame Relay switch with many ports: it supports connectivity between attached devices, but does not include complex operations like buffer management. Clouds allow you to model a subnetwork’s packet-loss

and latency characteristics while its inner workings remain transparent; see the *Standard Model Library* chapter of *Modeling Concepts* in the Modeler documentation or of the *User Guide* in the Guru documentation for more information.

Note—To add a Frame Relay demand object to your network model, open the Frame Relay object palette and drag the “fr_pvc” object to the workspace.

Note—If you do not specify a name for the PVC, the default name is *<source_endpoint_name>-<destination_endpoint_name>*.

IGRP

IGRP is the Interior Gateway Routing Protocol used in TCP/IP and OSI internets. It is classified under the interior gateway protocol (IGP) class of routing protocols, but can also be used as an exterior gateway protocol (EGP) for inter-domain routing.

Like the RIP routing protocol, IGRP uses distance vector routing technology. This technology specifies that gateways exchange routing information only with adjacent gateways. This routing information contains a summary of information about the rest of the network. The concept is that each router need not know all the router/link relationships for the entire network. Each router advertises destinations with a corresponding distance and the routers hearing the information adjust the distance and pass along the information to neighboring routers.

The distance information in IGRP is represented as a composite “metric” of available bandwidth, delay, load utilization, and link reliability. The dependence on all these metric-related parameters allows fine tuning of link characteristics to achieve optimal paths.

Model Features

The following features are implemented.

- The IGRP routing tables maintained at each node are initialized with the local gateway’s IP addresses. The cost for these routes is set to 0. The cost computation for routes learned from neighboring routers is performed using topological delay, bandwidth of the narrowest bandwidth segment, channel occupancy of the route, and the route reliability.
- *Equal-cost multi-path support*: In many cases it makes sense to split traffic between two or more paths. IGRP will do this whenever two or more paths are equally good.
- A modified Bellman-Ford algorithm (as documented in the reference above) is used during route computation.
- The start time at which the first regular routing update is generated is a parameter that you can control on a per-node basis.
- *Triggered updates* (that is, a new routing table) are sent every time a change is detected in the routing tables. Normally, new routing tables are sent to neighboring gateways on a regular basis (every 90 seconds by default; this can be adjusted using a model attribute).

- *Holddown timer* is implemented to avoid the problems due to potential latency in route propagation through the entire network. The holddown rule indicates that when a route is removed, no new route will be accepted for the same destination for some period of time. This holddown period ensures that triggered updates are delivered to all other gateways, so that any new routes received are indeed new, not an old route being re-inserted by a gateway that has not received the triggered update.
- *Split horizon with route poisoning* is implemented to avoid routing loops. Split horizon should prevent loops between adjacent gateways. Route poisoning is intended to break larger loops. The rule is that, when an update shows the metric for an existing route to have increased sufficiently, there is a loop. The route should be removed and put into holddown.
- *IGRP simulation efficiency*. With this technique (enabled by default), routing update messages are not sent after a certain point in time, resulting in faster simulation execution. If you know that routing information will not change after a certain point, enter that time as the IGRP Stop Time.

It is intended that only nodes that are gateways (those that have more than one network interface) implement IGRP. Therefore the IGRP routing tables are only used to route packets between gateways. This is important in order to reduce unnecessary simulation events and therefore improve simulation efficiency.

The IGRP route metric computation requires particular units for the bandwidth and delay components. These values are specified for each media type in a common external file (igrp_metric_compute_support.gdf):

Figure 1-1 Units for Bandwidth and Delay Components

```
# This file contains default values used for different media type for computing
# advertized route metrics. Most of the default values have been chosen from:
#
#   "An Introduction to IGRP" - by Charles L. Hendrick
#
# You may want to add more entries if you are modeling data rates not
# defined in this file. If a new entry is not added, the closest-lower data
# rate will be assumed for computing metric components.

# ATM:
622000000,              30
155000000,              40

# Ethernet:
1000000000,             20
100000000,              50
10000000,               100

#PPP
1544000                 2000

# FDDI:
100000000,              50

# Token Ring:
16000000,               90
4000000,                200
```

IP (Internet Protocol)

The Internet Protocol (IP) is a connectionless network-level protocol that interconnects networks. In the IP model suite, IP services transport layer protocols such as TCP and UDP. As such, IP is situated beneath the transport layer. In turn, IP relies on data link layer technologies, such as Ethernet and Token Ring, to relay packets to other IP modules. This document describes key features of the IP model shipped as part of the standard OPNET model library.

Model Features

This section provides a list of the main features available in the Internet Protocol model:

- The IP model suite captures the following protocol behavior:

Table 1-7 IP Model Features (Part 1 of 2)

Feature	Description
IP Addressing	IP addresses are specified using dotted decimal notation (also called dotted quad notation). The model incorporates addresses from class A, B, C, and D types of addresses. Class D addresses are used for IP Multicasting, and is modeled in OPNET's specialized IP Multicasting model.
Routing and Forwarding	IP routers are modeled using a finite buffer with two available processing modes: Slot-based and Central. Routing in the IP model is provided by: • dynamic routing tables — created by dynamic routing protocols • static routing tables — user-configured
Fragmentation and Reassembly	The IP model uses the same principles and methods as actual IP implementations to break up datagrams before transmission and to reassemble them at the destination IP module. This is essential if the model is to produce the same traffic characteristics that occur in the modeled network. Fragmentation occurs when the length of a datagram forwarded by an IP module exceeds the maximum transfer unit (MTU). Reassembly functions are performed on datagrams when they reach their final destination.

Table 1-7 IP Model Features (Part 2 of 2)

Feature	Description
Processing Delay and Queuing Capacity	The performance of IP-based networks depends on the characteristics and capabilities of the network's switching elements. To support accurate studies involving these parameters, the IP model allows you to specify queuing capacities and packet processing speeds for each IP module.
Queuing Algorithms	The model offers several queuing management options: • a standard FIFO queue • a weighted fair queue (WFQ) — Implemented as a bitwise round-robin fair queuing algorithm, it simulates the WFQ found in most routers. • priority queuing (PQ) • custom queuing (CQ) • random early detection (RED) • weighted random early detection (WRED)
IP Cloud Models	The standard model library includes IP cloud models that allow you to abstract parts of a network's infrastructure. A cloud behaves like a simplified IP-based router with many ports: it supports connectivity between attached devices, but does not include complex operations like WFQ. Clouds allow you to model only packet-loss and latency characteristics of a subnetwork. See for more information.
End of Table 1-7	

Reference Documents

The model is based on information from the following sources:

Table 1-8 Reference Documents

RFC-791	Internet Protocol
Parekh, A.K.J, and Gallager, R.G.	*A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: the Single Node Case*, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, January 1991
End of Table 1-8	

IP Addressing

An IP address is a 32-bit number consisting of two fields: the *network number* (identifying the network) and the *host number* (identifying a particular host within the network). An IP address is specified using dotted decimal notation. This notation uses four decimal integers separated by decimal points, where each integer is the value of one octet (8 bits) of the IP address. For example, the following 32-bit address:

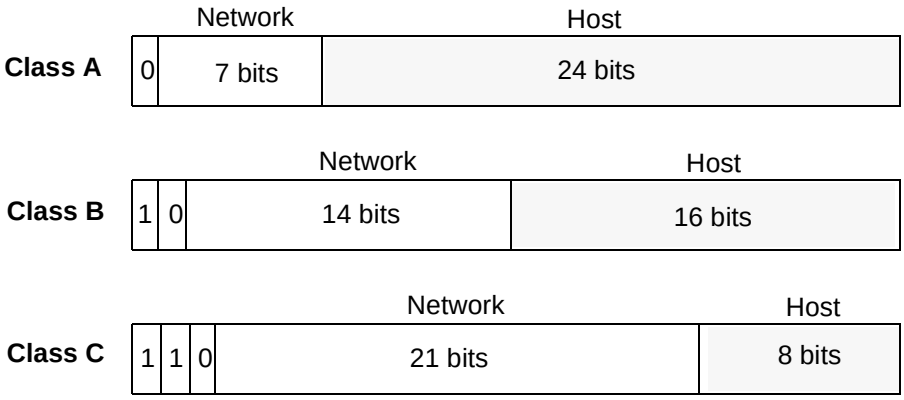
11000000 00001001 11001000 01101100

is written as:

192.9.200.108

Thus, an IP address is represented using the E.F.G.H format, where E, F, G, and H are decimal numbers between 0 and 255. Each of these decimal numbers represents one byte of the IP address. Together, the four bytes (32 bits) represent the network and host address. However, which numbers refer to the network and which numbers refer to the host depends upon the Internet address class. The following illustration highlights the most-commonly used address classes:

Figure 1-2 Internet Protocol Address Classes



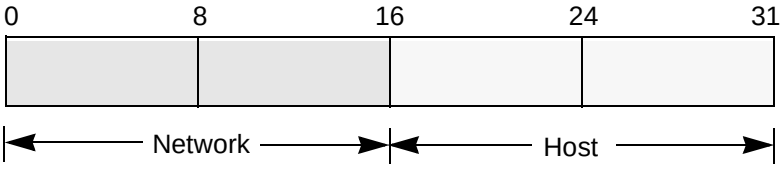
You can differentiate among the IP address classes by looking at the three most significant bits of the address. For example, the range of values for the first byte of Class C addresses begins at 192 (in decimal notation). The following table lists the range of values available as valid addresses for Class A, B, and C networks, based on the reservation of bits for network and host identifiers.

Table 1-9 Internet Protocol Address Range

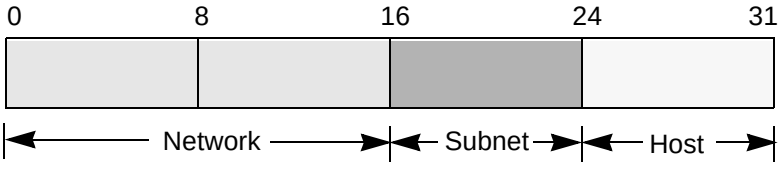
Class	Size (bytes)		Range of Values (dotted quad)	
	Network	Host	Network Portion	Host Portion
A	1	3	1 – 126	0.0.1 – 255.255.254
B	2	2	128.1 – 191.254	0.1 – 255.254
C	3	1	192.0.1 – 223.255.254	1 – 254
End of Table 1-9				

An integral part of IP addressing is *subnet addressing* (also called subnet routing or subnetting). Subnetting allows one IP network address to define and identify multiple underlying physical networks. This feature helps solve the potential problem of running out of the finite number of available IP network addresses. Subnet routing is achieved using extra bits from the host portion of the IP address. For example, consider a class B IP address: since it has 16 host number bits, it can assign 65536 node numbers. However, network configuration may require that nodes within this network are placed on different physical networks (for efficient administration, perhaps). If each physical network contains no more than 255 nodes, the higher order 8 bits of the host number portion of the IP address can be used for subnet addressing. Thus, the IP address representation changes from:

Figure 1-3 Internet Protocol Subnet Addressing



to:



This representation allows a maximum of 256 physical networks, each with 255 nodes, to be addressed with the single IP address.

A *subnet mask* identifies how IP nodes interpret IP addresses. When configuring a subnet mask, the number of extra bits used as the subnet number is determined. Then, a value of 1 is set for every bit position in the IP address that will be recognized as part of the network or subnetwork number. Additionally, a value of 0 is set for every bit position in the IP address that should be recognized as the host identifier (or node number). For example, if the eight most significant bits from the host identifier field of a class B address are to be used as the subnet numbers, the resulting subnet mask will be 255.255.255.0. For more details on IP addressing or subnetting procedures, see any standard IP textbook, or to the *Internet Engineering Task Force* document *RFC-950.*

IP Multicast

IP Multicast is an efficient way to distribute information from a single source to multiple destinations. The model library includes a highly detailed model for simulating IP multicast networks. This document describes key features of the IP Multicast model shipped as part of the standard model library.

Model Features

The IP Multicast model supports the following protocol features.

Table 1-10 IP Multicast Model Features

Feature	Description
Rendezvous Point (RP) Selection	The model supports dynamic and static RP selection. Two types of dynamic RP selection are supported: Bootstrap and Auto-RP. Auto-RP is feature of Cisco routers, which is automatically supported on all Cisco router models.
Shortest Path Tree (SPT) Switchover	Multicast routing uses the PIM-SM (Protocol-Independent Multicast - Sparse Mode) multicast routing protocol. The operation mode can be changed from shared-tree (RP tree) to shortest path tree (SPT tree) dynamically (during a simulation).
Capacity Planning	The model supports capacity planning studies using flow analysis and the Capacity Planning module.
Traffic Engineering	You can use the IP multicast model for the following types of traffic engineering studies: • Dependency on RP position • Dependency on multicast routing configuration • Dependency on the IP unicast routing table
End of Table 1-10	

Reference Documents

The model is based on information from the following sources:

Table 1-11 Reference Documents (Part 1 of 2)

RFC-1112	Host Extensions for IP Multicasting
RFC-2236	Internet Group Management Protocol, Version 2

Table 1-11 Reference Documents (Part 2 of 2)

RFC-2362	Protocol Independent Multicast-Sparse Mode (PIM-SM)
Auto-RP	Cisco Product Documentation
internet draft draft_ietf_pim_sm_bsr_03.txt	Bootstrap Router (BSR) Mechanism for PIM Sparse Mode
End of Table 1-11	

IS-IS

Intermediate System-to-Intermediate System (IS-IS) is an OSI link-state routing protocol. Originally designed to route packets in an ISO connectionless network protocol (CLNP), IS-IS was later extended to support routing in IP networks, including support for mesh groups and traffic engineering (TE) extensions. This document describes key features of the IS-IS model shipped as part of the standard OPNET model library.

Model Features

This section lists the main features available in the Intermediate System-to-Intermediate System model suite.

The IS-IS model suite captures the following protocol behavior:

Table 1-12 IS-IS Model Features (Part 1 of 2)

Model feature	Description
Link state routing	Router node models running IS-IS build routing tables by obtaining link information from a global link-state database and by using Dijkstra's algorithm to find the shortest path to every destination in the domain.
IS-IS packets over the MAC layer	IS-IS packets are transmitted directly over the MAC layer. This is in contrast to other routing protocols whose packets are encapsulated in an IP datagram before being sent to the MAC layer.
Multiple processes	A router can have multiple IS-IS routing processes active at the same time. When multiple processes are running, routing information is exchanged among the processes through redistribution.
Two-level hierarchy	An IS-IS domain can have a two-level hierarchy with a single Level-2 area and multiple Level-1 areas. Routers in Level-1 areas exchange routing packets with other routers in the same Level-1 area. To communicate with a router in another area, data packets are sent to the nearest Level-2 router, which routes the packets at Level-2.
Traffic engineering	The IS-IS model supports Constraint Shortest Path First (CSPF) computations, which are used to set up MPLS Label Switched Paths (LSPs). When setting up an MPLS LSP, the MPLS module requests a CSPF path from IS-IS or OSPF. If both protocols are enabled, the IS-IS path is used. The IS-IS TE feature consists of: • Dissemination of TE attributes in IS-IS link state packets using TLVs • CSPF computations to determine a path for an LSP The CSPF computation using IS-IS TE considers • metrics such as TE metric, IS-IS metric and delay • TE attributes such as available link bandwidth • include and exclude colors (resource classes)
Virtual links	IS-IS allows automatic repair of partitioned Level-1 areas using a virtual link through level-2. A partitioned level-2 cannot be repaired.
Hello protocol	IS-IS routers establish neighbor relationships by exchanging periodic hello messages with neighbors.

Table 1-12 IS-IS Model Features (Part 2 of 2)

Model feature	Description
Pseudo-nodes	To reduce the amount of routing information maintained in each router's link-state database, IS-IS represents broadcast networks as pseudo-nodes (PSN), where each router connected to the broadcast network has a link to the PSN and the PSN has a link back to each router.
Metric configuration	The metric (interface cost) is specified as an absolute number or it can be auto-calculated based on the interface bandwidth. Standard 6-bit and wide 3-byte metrics are supported.
Failure recovery	The IS-IS model suite supports failure and recovery of nodes and links in the network. The model learns of failures and recoveries through interrupts and modifies the global link-state database accordingly.
Overload	A router can indicate to other routers that its LS database is full by setting an overload flag in its packets. This will cause other routers to avoid using the overloaded router in their routing table calculations.
End of Table 1-12	

Model Limitations

The following features are not included in the model suite. As model development continues, many of these features will be implemented.

- Overload bit
- Virtual links
- Mesh groups
- IPv6 support

Reference Documents

The model is based on information from the following sources:

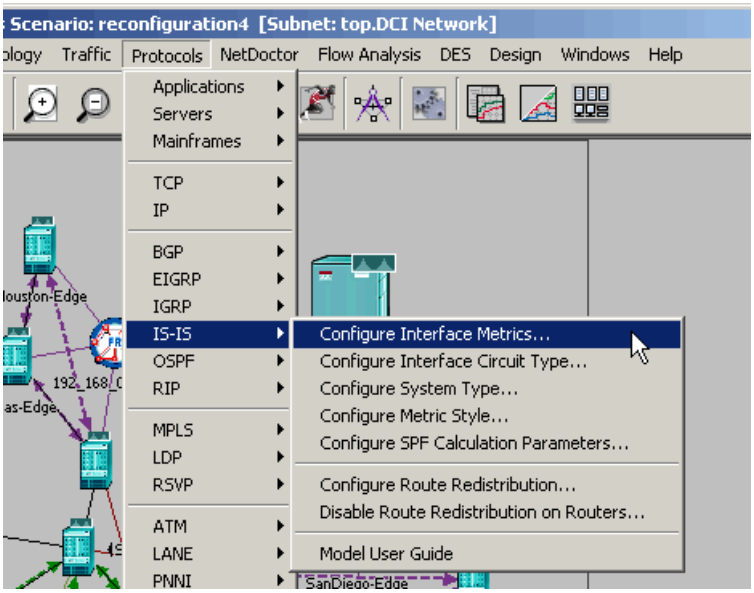
Table 1-13 Reference Documents

Document Description	Title
ISO 10589 (Original IS-IS specification for CLNP)	Information technology—Telecommunications and information exchange between systems—Intermediate system to Intermediate system intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode Network Service (ISO 8473)
RFC 1195 (IS-IS extension for IP)	Use of OSI IS-IS for Routing in TCP/IP and Dual Environments
IETF draft (TE extensions)	IS-IS extensions for Traffic Engineering draft-ietf-isis-traffic-04.txt
RFC 2973	IS-IS Mesh Groups
End of Table 1-13	

Global IS-IS Configuration

If you plan to use standardized configurations across the network model for IS-IS, you can save time by setting parameters from the Protocols menu.

Figure 1-4 Protocols Menu - IS-IS



The following table describes the configuration options available from the Protocols menu:

Table 1-14 Protocols Menu Options

Menu Selection	Description
Configure Interface Metrics...	Configure IS-IS interface metrics with the specified value. Apply to • Level-1 and/or Level-2 routers • Subinterfaces • Interfaces on selected links • All interfaces
Configure Interface Circuit Type...	Configure the IS-IS circuit type on • All interfaces, with or without subinterfaces • All connected interfaces • Interfaces on selected links
Configure System Type...	Specify the system type (Level-1, Level-2, or Level-1-2) on all routers in the network model or a selected set.
Configure Metric Style...	Specify the metric style (Narrow, Wide, or Both) on all routers in the network model or a selected set.
Configure SPF Calculation Parameters...	Configure the frequency at which the shortest-path first (SPF) calculations are performed. Set the values for all routers or a selected set.
Configure Route Redistribution	Control the way advertised routes from certain protocols are redistributed into IS-IS • Enable or Disable redistribution • Set metric • Apply to all routers or a selected set
Disable Route Redistribution on Routers...	Disable redistribution of advertised routes into IS-IS for all routers or a selected set.
Model User Guide	Opens this manual.
End of Table 1-14	

Link Aggregation

The OPNET link aggregation model implements the Link Aggregation Control Protocol (LACP) specified in section 43 of the IEEE 802.3-2002 standard. Cisco's EtherChannel is a form of link aggregation. In link aggregation, two or more Ethernet links of the same speed that connect a pair of nodes are aggregated to form a link aggregation group. These groups are treated as a single link by MAC client layers such as spanning tree protocols. LACP performs conversation-based load sharing among the links of the group.

Without link aggregation, multiple links between a pair of nodes does not increase the available bandwidth between the nodes. The spanning tree algorithm blocks the ports connected to the extra links to prevent routing loops. When links are aggregated by LACP, the spanning tree algorithm sees a single link with the combined bandwidth. LACP distributes the traffic among the component links using a configurable algorithm. Because LACP provides a single logical link to its clients, the spanning tree is not susceptible to the failures of individual links within the aggregated link. The spanning tree is only impacted if all of the member links fail (or, if all of the links are down, when one of them recovers).

Figure 1-5 Aggregating Links Between Two Nodes

Link aggregation is supported for router interfaces and switch ports. The procedure for configuring link aggregation differs slightly depending on the type of node (router or switch).

The model will bundle (aggregate) the ports or physical interfaces of a node if all of the following are true:

- The ports belong to the same group or the physical interfaces belong to the same aggregate interface
- The ports or physical interfaces have the same configuration
- The ports or physical interfaces are connected to the same node
- The corresponding ports or interfaces on the other node can be aggregated to each other

MPLS

Multi-Protocol Label Switching (MPLS) is a multi-layer switching technology that uses labels to determine how packets are forwarded through a network. SP Sentinel includes several complementary technologies that you can use to model and analyze MPLS networks.

Model Features

This section contains a list of the main features available in the Multi-Protocol Label Switching model: The MPLS model captures the following protocol behavior:

Table 1-15 MPLS Model Features

Feature	Description
LSP (Label Switched Path) configuration	• LSPs can be created manually or automatically from traffic conversation pairs. • LSPs are easily reused in other scenarios or projects by using the LSP import and export features. • Both dynamic and static LSPs are created using the path object.
Differential Services (DiffServ)	• DiffServ extensions, as defined in RFC-2475, are provided. • The model enables you to perform QoS (quality of service) analyses by accounting for different types of service.
Traffic Engineering	Traffic engineered routes are computed using Constrained Shortest Path First (CSPF) with OSPF or IS-IS routing protocols.
End of Table 1-15	

Table 1-16 Summary of MPLS Support in SP Sentinel

Technology	Supports MPLS	Special Requirements
Device Configuration Import	✔	Requires SP Guru or SP Sentinel.
Import from VNE Server	✔	Requires SP Guru or SP Sentinel.

Table 1-16 Summary of MPLS Support in SP Sentinel

Technology	Supports MPLS	Special Requirements
NetDoctor	✔	None.
Flow Analysis	✔	Requires SP Guru or SP Sentinel.
Discrete event simulation	✔	Requires MPLS specialized model license.
End of Table 1-16		

Reference Documents

MPLS models are implemented based on information available from the following sources.

Table 1-17 Reference Documents (Part 1 of 2)

Model Features	Document
Traffic Engineering	
MPLS TE	RFC-2702—Requirements for Traffic Engineering Over MPLS
FECs	RFC-3031—Multiprotocol Label Switching Architecture
IGP shortcuts	draft-hsmit-mpls-igp-spf-00
Label Switched Paths	
Dynamic LSPs Static LSPs	RFC-3031—Multiprotocol Label Switching Architecture
LSP routing	
OSPF TE IS-IS TE	RFC-2676—QoS Routing and OSPF Extensions
Label distribution	
LDP	RFC-3036—LDP Specification
CR-LDP	RFC-3212—Constraint-based LSP Setup Using LDP
RSVP-TE	RFC-3209—RSVP-TE: Extensions to RSVP for LSP Tunnels
PP-VPNs	

Table 1-17 Reference Documents (Part 2 of 2)

A framework for layer-3 PP VPNs	RFC-2547—BGP/MPLS VPNs
BGP/MPLS VPNs	draft-ietf-ppvpn-framework-05
Quality of Service	
QOS Architecture	RFC-2475—An Architecture for Differentiated Services
MPLS Support of Differentiated Services	RFC-3270—Multi Protocol Label Switching draft-ietf-mpls-diff-ext-08
Restoration and Resiliency	
Fast reroute with bypass tunnels LSP protection with ingress backup	draft-ietf-mpls-rsvp-lsp-fastreroute-00
End of Table 1-17	

OSPF

The Open Shortest Path First (OSPF) protocol is an interior gateway protocol (IGP) used for routing in Internet Protocol (IP) networks. As a link state routing protocol, OSPF is more robust against network topology changes than distance vector protocols such as RIP, IGRP, and EIGRP. Two versions of OSPF are implemented, OPSFv2 for IPv4 routing and OSPFv3 for IPv6 routing. This document describes key features of the OSPF model shipped as part of the standard OPNET model library.

Model Features

The section lists the main features available in the OSPF model suite.

Table 1-18 OSPF Model Features (Part 1 of 3)

		Supported In	
Feature	**Description**	**v2**	**v3**
Link-State Routing	OSPF is a link-state routing protocol. Each router sends link-state advertisements (LSAs) to the other routers in its domain. Upon receiving link-state information, routers use the shortest path first (SPF) algorithm to calculate the shortest paths to all destinations.	Yes	Yes
Equal-cost/Multi-path Routing	When there are multiple paths of equal cost to a destination, OSPF includes all of these paths in the IP routing table. This allows load balancing across equal-cost paths to a given destination.	Yes	Yes
Classless Inter-Domain Routing (CIDR)	OSPF supports CIDR addressing, which is also known as Variable Length Subnet Mask (VLSM). In CIDR, an IP network can be divided into subnets of different sizes, providing extra network configuration flexibility.	Yes	Yes
Multiple Processes	Multiple OSPF processes can be configured on a router. The interfaces of a router can be configured to run one of the processes configured on that router. Each OSPF process builds its own link state database. The Reports > OSPF Link State Database attribute lets you export at different times during the simulation.	Yes	Yes

Table 1-18 OSPF Model Features (Part 2 of 3)

Feature	Description	Supported In	
		v2	v3
Multiple Areas	OSPF can operate within a hierarchy. OPNET models have the following area configuration options: Areas: An OSPF network (such as an autonomous system, AS) can be divided into several areas, where each area is typically a group of contiguous networks and attached hosts. Routing information from one area is disseminated to other areas by area border routers (ABRs), which are routers that belong to multiple areas. Backbone: The OSPF backbone is responsible for distributing routing information among areas. All areas must be connected to the backbone area, by an ABR or by a virtual link. Since the backbone is also an OSPF area, routers in the backbone area maintain routing information using the same procedures and algorithms that any area router uses to maintain routing information. By default, all routers are configured to belong to the backbone area.	Yes	Yes
Traffic Engineering (TE)	The OSPF model suite supports TE extensions, which are used to set up MPLS LSPs (label switched paths). The following TE features are supported: • Distribution of TE attributes (cost, color, and bandwidth) through LSAs (link state advertisements) • Computation of CSPF (Constrained Shortest Path First) routes for LSP path calculation using cost or delay as the optimization metric	Yes	No
Topology Database	Each router maintains a topological database with the collection of LSAs received from all other routers in the same area. ABRs maintain separate topological databases for each area, where a topological database is essentially an overall picture of networks in relationship to routers. Since routers within the same area share the same information, they also have identical topological databases.	Yes	Yes
Virtual Links	OSPF areas can be defined in such a way that some areas are not physically connected to the backbone area by an ABR. In this case, backbone connectivity must be restored through virtual links. A virtual link functions like a direct link configured between a backbone router and a router in the unconnected area. Virtual links are also used to connect discontiguous parts of the backbone.	Yes	No
Route Aggregation at Area Boundaries	This is used to condense routing information by reducing a group of routes to a single advertisement, reducing both the load on the router and the perceived complexity of the network.	Yes	Yes
Designated Router Election	The model includes configurable parameters for determining which router acts as the designated router within an	Yes	Yes

Table 1-18 OSPF Model Features (Part 3 of 3)

Feature	Description	Supported In	
		v2	v3
Metric Configuration	An interface's cost is determined by setting it explicitly or from a bandwidth specification.	Yes	Yes
Hello Protocol	When a router's interface becomes active, it uses the Hello protocol to learn about its neighbors (i.e., routers that have interfaces to a common network). A router also exchanges hello packets that act as keep-alives with its neighbors—these inform neighboring routers that it is still functional.	Yes	Yes
Route Redistribution	Routes can be learned from other OSPF processes, from other routing protocols (RIP, IGRP, EIGRP, and BGP), or from directly connected interfaces not running OSPF.	Yes	No
End of Table 1-18			

- Configurable parameters per–IP interface include the following:
  - — Process and area to which the interface belongs
  - — Type of the interface (point-to-point, broadcast or non-broadcast multiple-access)
  - — Protocol timers (hello, router dead interval, retransmission)
  - — Priority (used for designated router (DR) election)
  - — Neighbor list (for NBMA networks)
- Configurable per-process parameters include the following:
  - — Address summarization
  - — External route information (OSPFv2 only)
  - — Virtual link configurations, including its associated timers (OSPFv2 only)
  - — Route redistribution (OSPFv2 only)

Reference Documents

The models are implemented based on information from the following sources:

Table 1-19 Reference Documents

Document	Title
RFC-2328	OSPF Version2
RFC 2740	OSPF for IPv6
Textbook	OSPF Complete Implementation — J. Moy
Textbook	OSPF: Anatomy of an Internet Routing Protocol — J. Moy
Textbook	Routing TCP/IP (Volume I) — J. Doyle
End of Table 1-19	

PNNI

Private Network-to-Network Interface (PNNI) is a hierarchical ATM routing protocol that distributes information among switches and computes paths through the ATM network. This document describes key features of the PNNI model shipped as part of the specialized OPNET model library.

Model Features

This section provides a list of the main features available in the PNNI model.

The PNNI model suite captures the following protocol behavior.

Table 1-20 PNNI Model Features (Part 1 of 2)

Feature	Description
Flexibility	The model has many configurable parameters. The PNNI model allows you to configure the following: • Up to 104 hierarchical levels • External static routes to external PNNI domains • Routing metric — metrics can be administrative weights, cell transfer delay, or cell delay variation • Complex node parameters to govern the costs of traversal of logical nodes
Report generation	The model provides several ways to easily display a network's configuration information. The PNNI model allows you to print the following: • Node information, such as group IDs, peer group leader status, and node ID • Route information for the PNNI domain • Hierarchical addressing information • Topology Database on a node at various times during the simulation
NNI Signaling	NNI (Network Node Interface) signaling is used during call setup and call tear-down procedures.

Table 1-20 PNNI Model Features (Part 2 of 2)

Feature	Description
Topology Discovery in Fast-Mode PNNI	Topology information is instantaneously propagated to all nodes in the network. No latency is involved in updating changes in the network. Topology databases are maintained on a per-group basis instead of a per-node basis. Topology database is not subject to aging.
Topology Discovery in Explicit PNNI	Topology discovery is done in the explicit mode, where the Hello protocol first discovers the neighbor links and nodes. Then, database synchronization and flooding is done to propagate the topology information in the network. All changes in the network take time to propagate to other nodes in the network. Topology databases are maintained on a per-node basis. Topology database is subject to aging.
Peer Group Leaders	Peer group leaders are selected based on: 1) the peer group leader election algorithm 2) the peer group leader priority configured on each node in a peer group.
End of Table 1-20	

- OPNET's PNNI model follows the ATM forum's PNNI 1.0 specification.
- Djkstra's algorithm is used to compute shortest path routes through the network.
- PNNI models are implemented based on information available from the following sources:

Table 1-21 Reference Documents

P-NNI V1.0	PNNI 1.0 ATM Forums: Private Network-Network Interface Specification Version 1.0 (PNNI1.0)
End of Table 1-21	

RIP

The Routing Information Protocol (RIP) is a distance-vector (Bellman Ford) routing protocol that is intended for gateways to exchange routing information between IP-based networks. RIP is used to obtain routing information to destinations. Each route has a metric or cost associated with it, which is the hop count to the destination.

Model Features

Features incorporated in this model are:

- The RIP routing tables are initialized with the local gateway's IP addresses. The cost for these routes is set to 0.
- Silent RIP processes are modeled with a parameter that can be controlled by the user. Silent RIP processes do not send out routing update messages, and are normally used for hosts that do no act as network gateways.
- The start time at which the first regular routing updates are generated is a parameter that can be controlled by the user.
- Split Horizon with Poisoned Reverse is implemented to avoid including routes in updates sent to the gateway from which they were learned. Such routes are included in updates, but their metrics are set to infinity.
- Regular and Triggered Updates
- Garbage Collection (Flush) and Timeout (Route Invalid) timers
- RIPv2

Unimplemented Features

The RIP example model provides most of the functionality of RIP as described in RFC 1058; however, the following feature is omitted:

- Specific routing update requests to ask for a gateway's entire or part of the routing table are not implemented. Requests are in most cases unnecessary and redundant, as the routing tables are propagated through periodic and triggered updates, as discussed above.

Reference Documents

The model is based on information from the following sources:

Table 1-22 Reference Documents

Document	Title
RFC 1058	Routing Information Protocol
RFC 2453	RIP Version 2
End of Table 1-22	

RSVP

Resource reSerVation Protocol (RSVP) is a network layer signaling protocol that allows applications to reserve network resources for unicast and multicast data flows. RSVP is used by:

- Transmitting applications to describe their data traffic characteristics
- Receiving applications to describe their Quality of Service (QoS) requirements
- Routers to deliver QoS requests to other routers along the path of a flow

Reference Documents

The model is based on information from the following sources.

Table 1-23 Reference Documents

RFC - 2205	Resource ReSerVation Protocol (RSVP) - Version 1 - Functional Specification
RFC - 2209	RSVP Message Processing
RFC - 2211	Specification of the Controlled-Load Network Element Service
End of Table 1-23	

Spanning Tree Bridge

The OPNET Spanning Tree Bridge (STB) model implements the spanning tree algorithm, which allows bridges to dynamically discover a loop-free subset of the topology.

Model Features

The OPNET bridge models support interfacing with Ethernet protocol both in a bus and hub configuration, as well as with Token Ring and FDDI protocols in a hub configuration.

Features incorporated in the bridge models are:

- Configuration of a Spanning Tree based on the network topology.
- Reconfiguration of the Spanning Tree based on topology changes (such as, link or node failures).
- Discarding of packets whose queuing delay within the bridge exceeds one second.

Method of Lazy Evaluation

This method is used in two cases in the OPNET bridge model to decrease the number of simulation events and therefore increase simulation efficiency. The Filtering Database contains entries for MAC addresses, which are learned when frames arrive at the bridge ports. These entries are timed out and deleted when their timer expires. Modeling this would require an interrupt for every single entry. In order to prevent this, every time an entry lookup is done in this database, the entry is evaluated to see whether it has expired. If expired, the entry is deleted; otherwise, the packet is routed according to the spanning tree algorithm.

A similar process is followed for removing packets which have exceeded a certain delay value at the port queues. When a packet arrives for service, its time spent in the queue will be evaluated. If this exceeds the recommended delay, the packet is discarded. This will prevent having timer interrupts for each packet in every port queue of the bridge.

Reference Documents

The model is based on information from the following source:

- IEEE Standard 802.1D, 1993 Specification: Local area networks — Media access control (MAC) bridges.

Vendor Device Models

All OPNET simulation products provide a rich library of pre-built vendor device models that you can use to build network models. Most were developed in response to client requests, with information collected from product catalogs and manufacturers' web sites.

Architecturally, certain device models do not contain internal details that the various vendor implementations offer, as most of these are vendor-proprietary. However, the models closely model the performance aspects of the modeled devices. These devices have been created using information obtained from publicly available device literature published by corresponding vendors. No other device-specific information has been provided by the device manufacturers or is represented in these models.

If you want to build a device model not present in the vendor model suite, there are two easy approaches:

- Use the Device Creator: This utility allows you to create several different types of network components -- routers, bridges, hubs, LAN nodes, multi-homed devices, switches, and vendor-specific devices. For more detailed documentation on how to use this utility, see the *Creating new devices with Device Creator* section of the Building Networks chapter of the User Guide manual in the Guru documentation or the Project Editor chapter of the Editor Reference manual in the Modeler documentation. Additionally, this document also provides a table listing the device types and protocols supported by Device Creator.
- Use Modeler to manually create devices: Leveraging the extensive standard protocol model library and Modeler's Node Editor, you can quickly and easily develop custom device models. Models created in Modeler can then be immediately used (without any modification) with IT Guru or SP Guru.

Model Features

This section discusses the implementation choices made in developing these models.

- All models implement "store-and-forward" switching (i.e., the reception of the complete packet from the incoming interface is awaited before starting to forward it on the outgoing interface).
- "Cut-through" switching is not modeled. This is because most bridges wait until the full packet has been received before attempting to forward it. The most noticeable difference between cut-through and store-and-forward architectures is that the former yields shorter latency.
- The models implement multiprocessor forwarding/routing if the represented device deploys these technologies.

- Switches and bridges model only layer 2 switching. IP layer routing is modeled in router models only.
- All bridge models, and the Ethernet and Token Ring switch models, are based on the IEEE Spanning Tree Bridge (STB) protocol. See Chapter 25 Spanning Tree Bridge Model User Guide on page STM-25-1 for more details. Some switch models also contain LANE support.
- Router models are IP-based.

Available Devices

The following devices are available as part of the Vendor Model suite.

Table 1-24 Vendor Device Model Library (Part 1 of 9)

Vendor Name	Product Family	Device Name	Device Type
3Com	CoreBuilder	CoreBuilder 2500	Layer-3 switch (modeled as high-speed router)
		CoreBuilder 3500	Ethernet switch
		CoreBuilder 6000	Layer-3 switch (modeled as high-speed router)
		CoreBuilder 7000	Ethernet switch with LANE interfaces
		CoreBuilder 7000HD	Ethernet switch with LANE interfaces
		CoreBuilder 9000	Enterprise switch (modeled as high-speed router)
	NetBuilder II	NetBuilder II	Enterprise Router (modeled as high-speed router)
	SuperStack II	SuperStack II Switch 1100	Ethernet switch
		SuperStack II Switch 3300	Ethernet switch
		SuperStack II Switch 1100 & 3300	Ethernet switch
		SuperStack II Switch 3800	Ethernet switch
		SuperStack II Switch 3900	Ethernet switch
		SuperStack II Switch 9000	Ethernet switch
		SuperStack II Switch 9300	Ethernet switch
ADC	Cuda	Cuda 12000	Cable Modem Termination System (CMTS)
AMD	ASUS	A7V Motherboard	Server
	Gigabyte	GA-7DX Motherboard	Server
	Tyan	Thunder K7 Motherboard	Server
Avici Systems	Terabit Switch Router	TSR	Internet Backbone Router

Table 1-24 Vendor Device Model Library (Part 2 of 9)

Vendor Name	Product Family	Device Name	Device Type
Cabletron	SecureFast	SFCS 200BX	ATM switch
	SmartSwitch	SmartSwitch 2040	Ethernet switch
		SmartSwitch 2200	Ethernet switch with LANE interfaces
		SmartSwitch 2208	Ethernet switch
		SmartSwitch 6000	Ethernet switch
		SmartSwitch 8000	Router
		SmartSwitch 8600	Router
		SmartSwitch 9000	Router
	SmartStack	SmartStack 10	Ethernet switch
		SmartStack 100	Ethernet switch
Cisco	Cat 1900 Series	Catalyst 1900	Ethernet switch
	Cat 2900 XL Series	Catalyst 2912 XL	Ethernet switch
		Catalyst 2916 MXl	Ethernet switch
		Catalyst 2924 XL	Ethernet switch
		Catalyst 2948G	Ethernet switch

Table 1-24 Vendor Device Model Library (Part 3 of 9)

Vendor Name	Product Family	Device Name	Device Type
Cisco (cont.)	Cat 3000 Series	Catalyst 3000	Ethernet switch
		Catalyst 3200	Ethernet switch
		Catalyst 3900	Token Ring switch
	Cat 5000 Series	Catalyst 5000	Ethernet switch
		Catalyst 5500	Ethernet switch
	Cat 6000 Series	Catalyst 6509	Router
	Cisco 1000 Series	Cisco 1005	Router
	Cisco 1600 Series	Cisco 1601	Router
		Cisco 1602	Router
		Cisco 1603	Router
		Cisco 1604	Router
		Cisco 1605R	Router
	Cisco 1700 Series	Cisco 1720	Router
		Cisco 1750	Router
	Cisco 2500 Series	Cisco 2501	Router
		Cisco 2502	Router
		Cisco 2509	Router
		Cisco 2510	Router
		Cisco 2511	Router
		Cisco 2512	Router
		Cisco 2513	Router
		Cisco 2514	Router
		Cisco 2515	Router
		Cisco 2524	Router

Table 1-24 Vendor Device Model Library (Part 4 of 9)

Vendor Name	Product Family	Device Name	Device Type
Cisco (cont.)	Cisco 2600 Series	Cisco 2620	Router
	Cisco 3600 Series	Cisco 3620	Router
		Cisco 3640	Router
		Cisco 3660	Router
	Cisco 3800 Series	Cisco 3810	Router
	Cisco 4000 Series	Cisco 4000	Router
		Cisco 4700	Router
	Cisco Access Server Series	Cisco AS4500	Access Server
		Cisco AS5100	Access Server
		Cisco AS5200	Access Server
		Cisco AS5300	Access Server
	Cisco 7000 Series	Cisco 7000	Router
		Cisco 7204	Router
	Cisco 7500 Series	Cisco 7505	Router
		Cisco 7507	Router
		Cisco 7513	Router
	IGX 8400 Series	Cisco 8420	Multiservice switch
	Cisco 8500 Series	Cisco 8540	Router
	BPX 8600 Series	Cisco 8650	ATM switch
	Cisco 12000 GSR Series	Cisco 12008	Router
		Cisco 12012	Router
		Cisco 12016	Router
	LS 1010 Series	LightStream 1010	ATM switch

Table 1-24 Vendor Device Model Library (Part 5 of 9)

Vendor Name	Product Family	Device Name	Device Type
Compaq	Alphaserver	AlphaServer 4100 5/533	Server
		AlphaServer DS10 6/600	Server
		AlphaServer DS20 6/500	Server
		AlphaServer DS20E 6/667	Server
		AlphaServer DS20E 68/833	Server
		AlphaServer DS20L 68/833	Server
		AlphaServer ES40 6/500	Server
		AlphaServer ES40 6/667	Server
		AlphaServer ES40 6/833	Server
		AlphaServer ES45 Model 68/1000	Server
		AlphaServer GS160 6/731	Server
		AlphaServer GS160 Model 16 68/1001	Server
		AlphaServer GS320 6/731	Server
		AlphaServer GS320 Model 32 68/1001	Server
		AlphaServer GS80 6/731	Server
		AlphaServer GS80 Model 8 68/1001	Server
Extreme Networks	Summit	Summit24	Router(Ethernet) switch
		SummitGbX	Ethernet switch
	Alpine	Alpine 3804	Router (Ethernet) switch
	BlackDiamond	BlackDiamond 6808	Router (Ethernet) switch
Fore Systems (Marconi Systems)	Cellpath	Cellpath 300	ATM Multiplexer

Table 1-24 Vendor Device Model Library (Part 6 of 9)

Vendor Name	Product Family	Device Name	Device Type
Fore Systems (Marconi Systems) (cont.)	ForeRunner	ASX-1000	ATM switch
		ASX 1200	ATM switch
		ASX-200BX	ATM switch
		ASX-200WG	ATM switch
		ASX-4000	ATM switch
		ForeRunnerLE 155	ATM switch with LES and BUS
		TNX-210	ATM switch
		TNX-1100	ATM switch
	ES	ES-3810	Ethernet switch
		ES-3810	Ethernet switch with LANE interfaces
		ES-3850	Ethernet switch with LANE interfaces
		ES-4810	Ethernet switch
		ESX-2400	Ethernet switch with LANE interfaces
		ESX-3000	Ethernet switch with LANE interfaces
		ESX-4800	Ethernet switch with LANE interfaces
	PowerHub	PH 6000	LAN switch
		PH 7000	LAN switch
		PH 8000	LAN switch
	TS	TS-2800	Token Ring switch with LANE interfaces

Table 1-24 Vendor Device Model Library (Part 7 of 9)

Vendor Name	Product Family	Device Name	Device Type
Foundry Networks	BigIron	BigIron 4000	Router (Ethernet) switch
		BigIron 8000	Router (Ethernet) switch
	FastIron	FastIron II	Ethernet switch
	NetIron	NetIron 400	Router (Ethernet) switch
		NetIron 800	Router (Ethernet) switch
Hewlett-Packard	AdvanceStack	AdvanceStack 10Base-T Hub 16U	Ethernet 10BaseT Hub
		AdvanceStack Switch 800T	Ethernet switch
	ProCurve	ProCurve 10Base-T Hub	Ethernet 10BaseT Hub
		ProCurve 100Base-T Hub	Ethernet 100BaseT Hub
		ProCurve Switch 212M	Ethernet switch
		ProCurve Switch 224M	Ethernet switch
		ProCurve Switch 1600M	Ethernet switch
		ProCurve Switch 2224	Ethernet switch
		ProCurve Switch 2424M	Ethernet switch
		ProCurve Switch 4000M	Ethernet switch
		ProCurve Switch 8000M	Ethernet switch
		ProCurve Routing Switch 9304M	Router
		ProCurve Routing Switch 9308M	Router
Juniper	Backbone Router	M20	Internet Backbone Router
	Backbone Router	M 40	Internet Backbone Router
	Backbone Router	M160	Internet Backbone Router

Table 1-24 Vendor Device Model Library (Part 8 of 9)

Vendor Name	Product Family	Device Name	Device Type
Lucent & Ascend	GRF	GRF 400	IP switch (modeled as high-speed router)
		GRF 1600	IP switch (modeled as high-speed router)
	MAX	MAX 4048	IP switch (modeled as high-speed router)
		MAX TNT	IP switch (modeled as high-speed router)
	STDX	STDX 6000	Frame Relay switch
	GX	Lucent GX 550	ATM switch
Lucent & Ascend (cont.)	CBX	Lucent CBX 500	Router
	S-TDX	Lucent S-TDX 8000	Router
		Lucent S-TDX 9000	Router
NEC	Media Gateway	CX3200 Media Gateway	Router
	IP Switch Router	CX5210	Router
Newbridge	MainStreetXpress	MainStreetXpress 36035	Router
		MainStreetXpress 36075	Router
	VIVID	VIVID CS1000	ATM switch
		VIVID CS3000	ATM switch
		VIVID GeoRimE24	Ethernet switch
		VIVID GeoRimFE	Ethernet switch
		VIVID GeoStaxE16	Ethernet 10BaseT Hub
		VIVID GeoStaxE24	Ethernet 10BaseT Hub

Table 1-24 Vendor Device Model Library (Part 9 of 9)

Vendor Name	Product Family	Device Name	Device Type
Nortel & Bay Networks	Accelar	Accelar 790	Load Balancer
		Accelar 1050	Routing (Ethernet) switch
		Accelar 1100	Routing (Ethernet) switch
		Accelar 1150	Routing (Ethernet) switch
		Accelar 1200	Routing (Ethernet) switch
		Accelar 8010	Enterprise Ethernet switch
		Accelar 8100 (in development)	Enterprise Routing switch
		Accelar 8600 (in development)	Enterprise Routing switch
	Alteon 180 Series	Alteon 180e	Load balancer
	ASN	Access Stack Node	Router
	BCN	Backbone Concentrator Node	Router
	BLN	Backbone Link Node	Router
	BayStack	BayStack 350T	Routing (Ethernet) switch
		Bay Stack 450	Routing (Ethernet) switch
	Centillion	Centillion 100	Ethernet switch with LANE
		Centillion 100	Token Ring switch with LANE
	Contivity	1500	Access Router
	CVX	1800	Access Router
	Passport	Passport 7480	Multi-service switch
		Passport 15000	Multi-service switch
	Shasta	Shasta 5000	Broadband Service None
	UE	UE 9000	Multi-service Edge Router

End of Table 1-24

Devices That Can Be Created Using the Device Creator

Device Creator can create the following types of devices.

Table 1-25 Device Creator: Supported Device Categories (Part 1 of 2)

Device Class	Supported Protocols
Bridge	Ethernet (10BaseT, 100BaseT, 1000BaseX) FDDI Token Ring
Hub	Ethernet (10BaseT, 100BaseT, 1000BaseX) FDDI Token Ring
LAN Nodes	Ethernet (10BaseT, 100BaseT, 1000BaseX) FDDI Token Ring
Multihomed Hosts	ATM Ethernet (10BaseT, 100BaseT, 1000BaseX) FDDI Frame Relay LANE (ethernet and token-ring) SLIP Token Ring

Table 1-25 Device Creator: Supported Device Categories (Part 2 of 2)

Device Class	Supported Protocols
Router	ATM
	Ethernet (10BaseT, 100BaseT, 1000BaseX)
	FDDI
	Frame Relay
	LANE (ethernet and token-ring)
	SLIP
	Token Ring
Switch	ATM
	Ethernet (10BaseT, 100BaseT, 1000BaseX)
	FDDI
	Frame Relay
	LANE (ethernet and token-ring)
	Token Ring
	Multi-protocol
Vendor Devices	3Com
	Ascend (Lucent)
	Avici Systems
	Bay Networks (Nortel)
	Cabletron
	Cisco Systems
	Extreme Networks
	Fore Systems
	Foundry Networks
	Hewlett-Packard
	Juniper Networks
	Lucent (Ascend)
	NEC
	Newbridge Networks
	Nortel Networks
End of Table 1-25	

Model Naming Convention

The following format is used to name models in the Vendor Models suite.

<Vendor_Name>_<Device_Name>_<Number_of_Slots>_<Configuration>

<Vendor_Name> is the abbreviated name of the vendor whose device is modeled

<Device_Name> is the specific product name

<Number_of_Slots> is the number of slots used in the configuration of the modeled device

<Configuration> is the port configuration, by protocol, for the modeled device

Vendor Name Conventions

The following table presents the supported vendor names:

Table 1-26 Supported Vendor Names (Part 1 of 2)

Abbreviation	Vendor Name
3C	3Com
AS	Ascend
AV	Avici Systems
BN	Bay Networks
CB	Cabletron
CS	Cisco Systems
EX	Extreme Networks
FS	Fore Systems
FD	Foundry Networks
HP	Hewlett Packard
JN	Juniper Networks
LU	Lucent

Table 1-26 Supported Vendor Names (Part 2 of 2)

Abbreviation	Vendor Name
NB	Newbridge
NEC	NEC
NT	Nortel
End of Table 1-26	

Configuration Conventions

The following table presents the abbreviations used for interfaces in the names of the vendor models:

Table 1-27 Interface Abbreviations

Configuration	Interface description
a	ATM
ae	Auto-negotiate Ethernet (10BaseT/100BaseT/1000BaseX)
e	Ethernet (10baseT)
f	FDDI
fe	Fast Ethernet (100BaseT)
fr	Frame relay
ge	Gigabit Ethernet (1000BaseX)
Sl	SLIP[1]
tr	Token Ring
End of Table 1-27	

1. **S**erial **L**ine **I**nternet **P**rotocol **I**nterface.

Example Model Names

The following table decodes a few device model names. Models are also described in the Comments section of the Model Description dialog box.

Table 1-28 Vendor Model Naming Conventions

Model Name	Vendor	Device	Configuration
3C_CB6000_4s_e16_f2_fe9_tr8	3Com	CB6000	A CoreBuilder 6000 router chassis with four slots supporting 16 Ethernet (10BaseT) ports, two FDDI ports, nine fast Ethernet (100BaseT) ports, and eight token ring ports.
BN_Centillion100_3s_a8_e8_fe2	Bay Networks	Centillion100	A switch with three slots (switching host modules), supporting eight ATM ports, eight Ethernet (10BaseT) ports, and two fast Ethernet (100BaseT) ports.
End of Table 1-28			

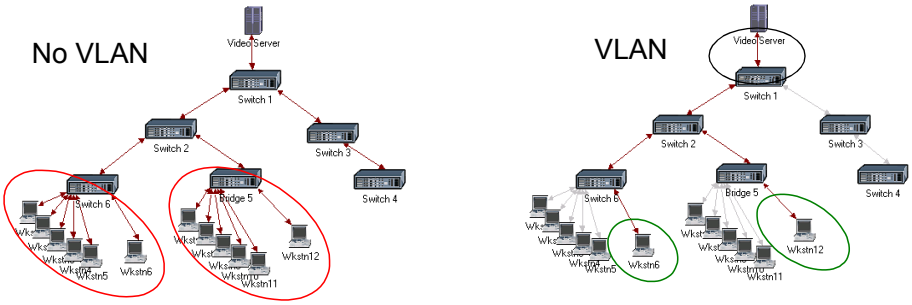
VLAN

A virtual LAN (VLAN) is a group of network users and resources, logically segmented by some function, such as a project team or application, without regard to the physical locations of the nodes. Unicast, broadcast, and multicast packets are forwarded and flooded only to stations in the VLAN. Each VLAN is considered a logical network, and packets moving from one VLAN to a different VLAN must be forwarded through a layer-3 device. Because VLANs are logically separate networks, each VLAN can support its own spanning tree over the switched network, independent of other VLANs.

VLANs have the following advantages:

- **Broadcast control:** Prevents traffic destined for a group of users (such as an IP multicasting group) from flooding the entire network.
- **Security:** Administrators have control over each port and resource.
- **Flexibility and scalability:** Physical location of nodes is no longer a factor.

Figure 1-6 Using VLANs for Broadcast Control



In a broadcast from the video server, all workstations receive traffic.

With a VLAN defined to include workstations 6 and 12, only VLAN members receive traffic.

This document describes key features of the VLAN model shipped as part of the standard OPNET model library.

Model Features

The VLAN model suite captures the following protocol behavior:

Table 1-29 VLAN Model Features (Part 1 of 2)

Feature	Description	Related Attributes
RSM	Enables routing and switching in one device. An RSM (route switch module) node can be created using the Device Creator utility to build a switch that includes an IP layer for routing.	
Layer-2 Switches (Bridges)	Switches are either VLAN-aware or VLAN-unaware. If VLAN-unaware, switch ignores VLAN tags if they exist in the received packet. If VLAN aware, switch performs its forwarding task based on the ingress and egress rules defined in the IEEE 802.1Q standard.	VLAN Parameters Scheme Set to “Port-Based VLAN”
	If VLAN-aware, all supported VLANs must be defined for a switch. (If no VLANs are specified for a node, it supports only VLAN 1 by default.)	VLAN Parameters Supported VLANs
	If VLAN-aware, VLAN support must be defined for each port. Supported VLANs for a port must be a subset of supported VLANs for the switch.	Switch Port Configuration VLAN Parameters Supported VLANs
VLAN Tags	Untagged frames carry no VLAN membership info. VLAN-tagged frames have header tags containing a VLAN ID (VID).	
Ports (of VLAN-aware Switches)	Can support one or more VLANs. (If no VLANs are specified for a port, it supports only VLAN 1 by default.)	Switch Port Configuration VLAN Parameters Supported VLANs
	Determine VLAN membership for untagged frames by using the Port VLAN Identifier (PVID) of the receiving port. PVID must be configured. Default PVID is 1.	Switch Port Configuration VLAN Parameters Port VLAN Identifier
	Can be Access, Trunk, or Hybrid. Default port type is Access.	Switch Port Configuration VLAN Parameters Port Type
Access Ports	Connect end stations to VLAN-aware network. Send all frames untagged. If frame is VLAN tagged, they strip off the tag before sending. Expect to receive VLAN-untagged frames. Insert VLAN tag based on the PVID of the receiving port.	
Trunk Ports	Connect VLAN-aware nodes Send all frames VLAN-tagged. Expect to receive VLAN-tagged frames. Can add a VLAN tag based on the PVID of the receiving port, if frame is untagged.	

Table 1-29 VLAN Model Features (Part 2 of 2)

Feature	Description	Related Attributes
Hybrid Ports	Connect both VLAN-aware and unaware nodes. Send either VLAN-tagged or untagged frames.	
	Consult Supported VLANs table to determine whether to send VLAN-tagged or untagged. Send VLAN-tagged frames to VLAN-aware nodes and send VLAN-untagged frames to VLAN-unaware end stations.	Switch Port Configuration VLAN Parameters Supported VLANs > Tagging (for hybrid ports) Specify Send Tagged or Send Untagged.
VLAN Links	are defined by the ports they connect. Trunk links are connections between two trunk ports. They carry frames with VLAN tags. Access links are entry points to a VLAN-aware portion of the network. They are connected to devices that are unaware of any VLAN membership. Switches remove VLAN tags before transmitting on access links. Hybrid links transport packets either VLAN-tagged or untagged, based on the VLAN classification of the packet.	
Spanning Tree Protocol	You can configure the traffic of all VLANs to use a single spanning tree. Alternatively, the traffic of each VLAN can be mapped to a separate spanning tree by deploying the MSTP or PVSTP protocols. The switch performs Shared VLAN Learning or Independent VLAN Learning based on the spanning tree protocol selection.	VLAN Parameters Spanning Tree Creation Mode Specify Multiple, Shared, or Per-VLAN.
End of Table 1-29		

Reference Documents

The model is implemented based on information from the following sources:

Table 1-30 Reference Document

IEEE-802.1Q	Virtual Bridged Local Area Networks
End of Table 1-30	

2 System Environment

This chapter describes various aspects of the interface between the SP Sentinel system and its operating environment. These aspects include the operating system, directory structure, and file naming conventions.

Operating System

The SP Sentinel system operating environment typically resides on an engineering workstation. Sentinel has been separately ported to the native operating system (OS) of each supported hardware platform. The OS versions supported by SP Sentinel, along with other system requirements, are listed on the OPNET web site at http://www.opnet.com.

The Sentinel system consists of a set of well-behaved application programs, object code libraries, and data files. In this context, *well-behaved* means that the Sentinel components interact with the operating system and the user in a predictable manner that is common to other programs. For instance, Sentinel programs run as regular user processes and do not interfere with the operating system's control of the computer. In contrast, a non-well behaved program might contain its own operating system kernel and usurp control of the computer from its native operating system while it executes.

Whenever possible, the Sentinel system abides by and fosters the concept of simple tools with well-defined interfaces that can be combined by the user with other tools in creative ways.

File System Organization

For convenience, Sentinel files are organized into at least three general locations within the file system. These locations allow files of different types to be stored separately, as follows:

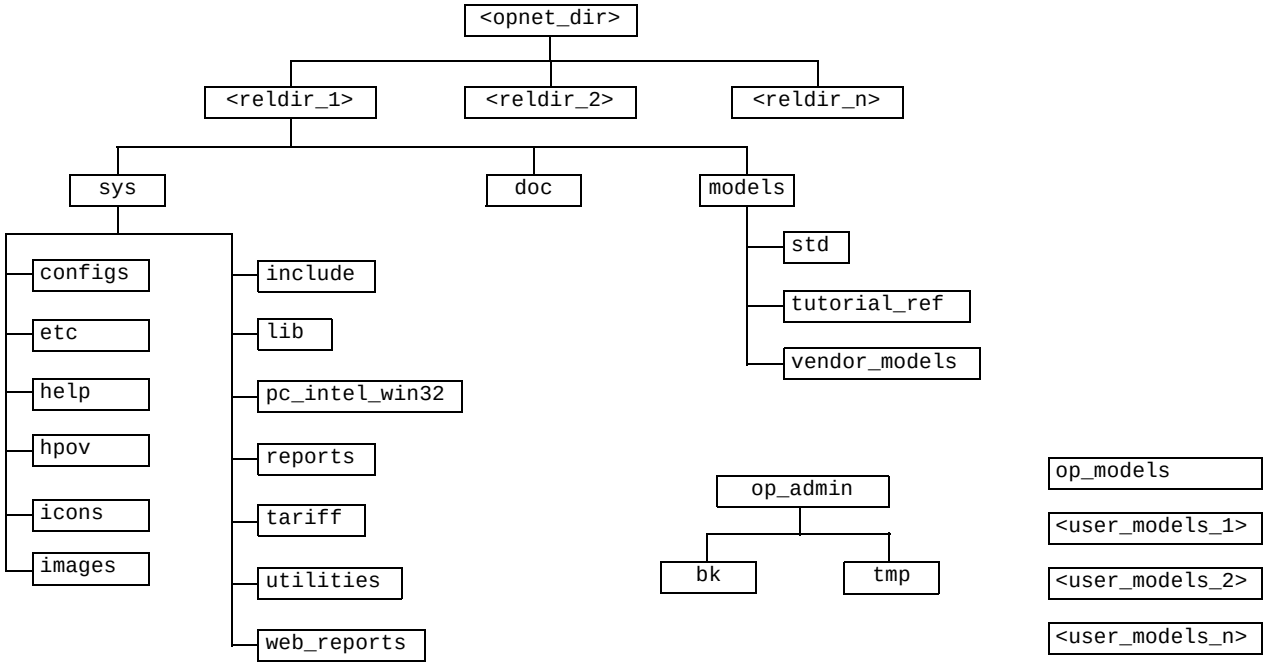
- The OPNET directory contains Sentinel system software and standard models. This directory is described in OPNET Directory (<opnet_dir>) on page SSR-2-4.
- The administration directory contains configuration and by-product files. This directory is described in Administration Directory (op_admin) on page SSR-2-12.
- One or more model directories (in one or more locations) contain user-developed models. These directories are described in section User Model Directories on page SSR-2-16.

Components of the licensing system are not stored in the OPNET directory, but locally, in one of the following locations:

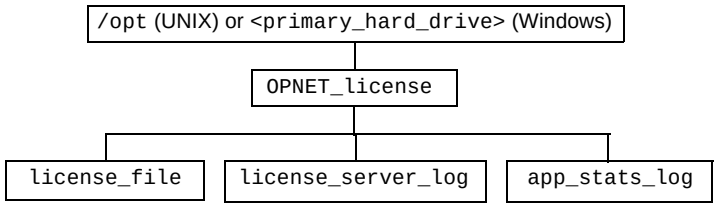
- `<primary_hard_drive>:\OPNET_license` (Windows platforms)

  The `<primary_hard_drive>` is the first drive letter (usually C:) that names a non-removable, non-network local drive partition.
- `/opt/OPNET_license` (UNIX platforms)

Figure 2-1 File System Organization



Components of the licensing system are stored locally



OPNET Directory (<opnet_dir>)

The OPNET directory (<opnet_dir>) is the directory where the Sentinel release media contents are downloaded. The OPNET directory is an important reference location on the disk. It is usually represented by the symbol <opnet_dir>, especially when it appears as part of a path name, as in <opnet_dir>/<release>/sys/include.

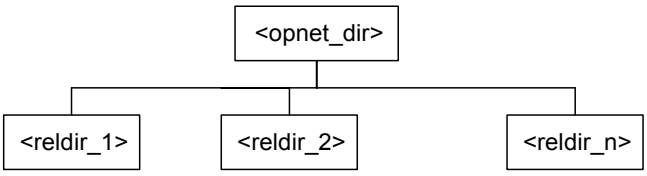
The symbol <opnet_dir> represents an actual directory located somewhere in the file system. The user (or system administrator) has the freedom to select the location of <opnet_dir> and create the directory there; no hardwired location is required by Sentinel. One of the typical locations for <opnet_dir> is in the /usr directory (for example, /usr/opnet). This location is beneficial when several users with different accounts need access to the SP Sentinel software. In this case, <opnet_dir> is owned by the root or system administrator account, with read and execute permissions set up for other accounts. Another common location for <opnet_dir> is within a user's home directory. This location is useful when only one user requires access to the Sentinel software and it is preferred to handle all system administration tasks locally. In this case, <opnet_dir> should not be given the name <HOME>/spsentinel, because this name may conflict with execution of the spsentinel program in the home directory. Instead, choose a neutral name such as <HOME>/opnet.

Because the OPNET directory stores the Sentinel system software (including software for different releases or different workstation architectures), there typically is only one OPNET directory per installation site. Even when several workstations are used to execute the Sentinel software, all the workstations can share one <opnet_dir> located on a file server (via the Network File System, NFS, or a similar system).

The OPNET directory contains only one type of subdirectory: <reldir>. The release directory contains the system software and standard models for a particular release of Sentinel. There is always at least one release directory.

When installed, new Sentinel releases replace the existing files within these two subdirectories, but the top-level organization remains unchanged. The following figure depicts this organization.

Figure 2-2 OPNET Directory Organization



Release Directory (<reldir>)

The release directory (<opnet_dir>/<reldir>) contains the complete software and model files for a specific Sentinel release. A *release* is a distinct version of Sentinel that typically incorporates new features or major bug fixes. A release directory is identified by a name such as 11.5.A.

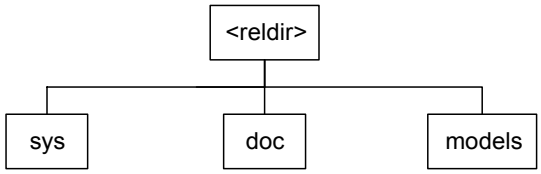
Each release directory contains three subdirectories:

Table 2-1 Contents of the Release Directory

This subdirectory...	Contains...
sys	the system software for the corresponding release
doc	data files used by the documentation viewer
models	the models supplied with Sentinel
End of Table 2-1	

The following figure depicts this organization.

Figure 2-3 Release Directory Organization



If you download a new release or software update into an existing <reldir> directory, it replaces the existing files within these two subdirectories, but the top-level organization remains unchanged. Typically, however, a new Sentinel release will go into a new <reldir> directory, in which case any existing <reldir> directories (and their contents) remain unchanged. The names and locations of subdirectories under each <reldir> directory should not be modified after installation.

System Directory (sys)

The system directory (<opnet_dir>/<reldir>/sys) contains the Sentinel software, including all programs, libraries, and header files. The organization of sys is copied from the Sentinel release media and remains fixed. The contents of sys subdirectories vary somewhat during installation and software updates, but remain fixed during use of the software (that is, no non-installation programs modify the contents of the sys directory and its subdirectories).

Because sys contains all of the Sentinel software, it is a resource that can be shared by multiple users. It is also possible that more than one version of Sentinel will be present at a site, due to different machine architectures. The software for different architectures is stored in individual <arch> subdirectories.

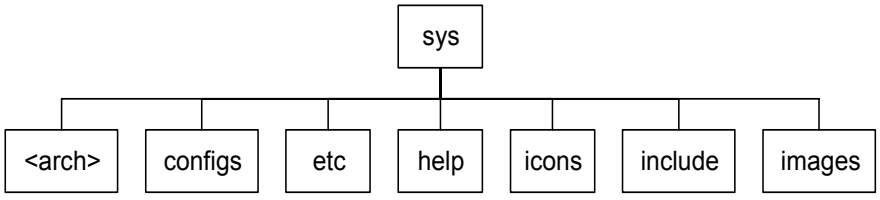
The sys directory contains these types of subdirectories:

Table 2-2 Contents of the sys Directory

This subdirectory...	Contains...
<arch>	architecture-specific system software. One of these subdirectories exists for each different workstation architecture in use.
include	header files used during the compilation of process models.
etc	miscellaneous files and utilities.
configs	configuration files.
icons	icon databases used by Sentinel and by specific GUI-based programs.
images	screen and web images used by Sentinel.
help	data files used by spsentinel's help documentation.
End of Table 2-2	

The following figure depicts this organization.

Figure 2-4 System Directory Organization



Architecture Directory (<*arch*>)

The architecture directory (<opnet_dir>/<reldir>/sys/<arch>) contains the program executables, object code libraries, and other information needed for a specific workstation architecture.

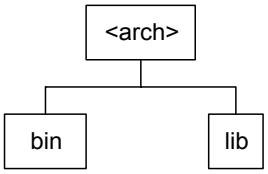
Each architecture directory contains one or two subdirectories:

Table 2-3 Contents of the <arch> Directory

This subdirectory...	Contains...
bin	program executables for the specified architecture.
lib	object code libraries that constitute the Simulation Kernel and GUI reference data files for the specified architecture.
End of Table 2-3	

The following figure depicts this organization.

Figure 2-5 Architecture Directory Organization



In addition, there is one special architecture directory called <opnet_dir>/<reldir>/sys/unix. This directory contains only a bin subdirectory, which contains scripts used under UNIX to set the LD_LIBRARY_PATH preference and to help Sentinel automatically locate the correct binaries for the workstation architecture being used.

Binary Directory (bin)

The binary directory (<opnet_dir>/<reldir>/sys/<arch>/bin) contains Sentinel programs, C shell scripts, and batch files. These include installation programs, model building programs, and special utilities.

Library Directory (lib)

The library directory (<opnet_dir>/<reldir>/sys/<arch>/lib) contains the Sentinel library and reference data files.

Include Directory (include)

The include directory (<opnet_dir>/<reldir>/sys/include) contains the Sentinel header files. Header files are C language files that contain symbolic constants, data structure definitions, and preprocessor macros. Each of the header files contained in include has the conventional ".h" suffix. Explanations of individual header files are not provided here because you do not need to individually reference these files.

Miscellaneous Directory (etc)

The miscellaneous directory (<opnet_dir>/<reldir>/sys/etc) contains miscellaneous files and utilities.

Configuration Directory (configs)

The configuration directory (opnet_dir>/<reldir>/sys/configs) contains files used to specify the configuration of the user interface, such as menu items and shortcuts. These files have the file extension ".ets".

Icon Directory (icons)

The icon directory (opnet_dir>/<reldir>/sys/icons) is the standard location for files containing icons used by GUI-based programs. (Another common location is each user's administration directory.) Icon databases can be stored anywhere, however, so long as you specify the location to Sentinel by including the icon database directory in the value of the mod_dirs preference.

Icon databases in directories listed earlier in mod_dirs override corresponding databases in directories specified later in the list. Typically, the system and program icon databases provided with Sentinel will remain in the icon directory, unmodified and unoverridden, and user's special icons will be stored in the user icon databases within their administration directories. In this case, the administration directory should appear before the icon directory in the mod_dirs list.

The icon_dbs preference specifies which icon databases a particular Sentinel application should use. GUI-based Sentinel programs use three types of icon databases:

Table 2-4 Types of Icon Databases

Icon database	Contents
sys.icons	The system icon database contains icons used in most GUI-based programs.
<program>.icons	Separate program icon databases exist for each GUI-based program (such as spsentinel, op_vuanim, and op_edmap). These databases contain icons specific to the corresponding program.
<name>.icons	User icon databases contain user-defined icons created with the Icon Editor.
End of Table 2-4	

In addition to the icons in these databases, a special “missing” icon is built into GUI programs. A program displays this icon when it can locate neither the specified icon (usually a user-defined icon) nor an appropriate default icon.

Image Directory (images)

The image directory (`opnet_dir>/<reldir>/sys/images`) contains bitmap image files used by the spsentinel program, including the splash screen and various images used by web reports.

Documentation Directory (doc)

The documentation directory (`opnet_dir>/<reldir>/doc`) contains files for the <product> documentation and help. These files contain the SP Sentinel printed documentation in online form, the Administrator Guide, tutorials, and menu files for selecting the desired information. These files are stored in Adobe Acrobat format. It also contains the necessary viewer software.

Models Directory (models)

The models directory (`opnet_dir>/<reldir>/models`) contains the Sentinel models that are included on the release media. The organization of models, like that of sys, is initially obtained from the Sentinel release media and generally remains fixed. The models directory usually contains five subdirectories:

Table 2-5 Contents of the models Directory

Subdirectory	Contents
std	The standard models directory contains OPNET-developed models of basic nodes and links, plus example network models.
tutorial_ref	The tutorial models directory contains a set of reference files showing correct results for models created from the tutorials.
contrib	The contributed model directory contains models created by other users and made available to the OPNET user community.
third_party	The third-party model directory contains models created by third-party vendors. These models must be obtained from the individual vendors.
compat	The compatibility model directory contains models from previous releases of Sentinel.
vendor_models	This directory contains device models based on many different sources, including studies conducted at the Harvard Network Device Test Laboratory and product data from the device makers.
End of Table 2-5	

You can use the models in the std and contrib subdirectories as guides or as a basis for developing your own network models.

Standard Models Directory (std)

The standard models directory (`opnet_dir>/<reldir>/models/std`) contains models developed by OPNET and shipped with Sentinel. These models include the basic nodes and links used to develop networks, as well as complete network models.

The organization of std is initially obtained from the Sentinel release media and generally remains fixed. The following sections describe the subdirectories of std and their contents.

Tutorial Reference Models Directory (tutorial_ref)

The tutorial models directory (`opnet_dir>/<reldir>/models/tutorial`) contains reference models for the tutorials. Reference models are completed examples of the files created in each lesson. The required models for the tutorial exercises are in the `opnet_dir>/<reldir>/models/std/tutorial_req` directory.

Contributed Models Directory (contrib)

The contributed models directory (`opnet_dir>/<reldir>/models/contrib`) contains models developed by other users and made available to the Sentinel user community free of charge. These models are not supported by OPNET Technologies, Inc., but might be useful in your modeling efforts. Each contributed model contains a text file describing its use. You can download the latest contributed models from the OPNET FTP site. Contact OPNET Technical Support for information on how to access these models.

Compatibility Models Directory (compat)

The compatibility models directory (`opnet_dir>/<reldir>/models/compat`) holds link and node models converted from earlier Sentinel releases. Because many models are modified from one release to the next, these converted models may be needed to allow older network models to run under a newer release. Additional details are available in the Release Notes for each release.

Vendor Models Directory (vendor_models)

The vendor models directory (`opnet_dir>/<reldir>/models/vendor_models`) holds models based on a variety of sources, including studies conducted at the Harvard Network Device Test Laboratories (HNDTL) and generally-available product data from the device makers.

Note—No proprietary information was provided by device manufacturers.

Licensing System Components

The licensing system consists of three separate data files:

- license_file contains the individual licenses allocated to users. See the Licensing chapter in the Administrator Guide for details.
- license_server_log contains the log of events kept by the license server. See the Licensing chapter in the Administrator Guide for details.
- app_stats_log contains the usage statistics displayed when you choose Show Local Server's Usage Statistics from the License Servers menu of the License Manager.

These files are stored locally, in one of the following locations:

- `<primary_hard_drive>:\OPNET_license` (Windows platforms)

  The `<primary_hard_drive>` is the first drive letter (usually C:) that names a non-removable, non-network local drive partition.

- `/opt/OPNET_license` (UNIX platforms)

Administration Directory (op_admin)

The administration directory (<HOME>/op_admin) contains configuration information and files specific to an individual Sentinel user, such as:

- program configurations
- model directory paths
- logical key definitions
- model backups

Because the OPNET directory serves as a centralized resource for all users at a site, it is not an appropriate place to store files that are specific to a particular user. Such files are best stored in the home directory of each user, where they can be easily found and are protected by the user's directory permissions. For this reason Sentinel supports individual *administration directories* in which users store their personal Sentinel files.

The administration directory must be named op_admin so that programs can access it via the shell variable-based path name `<HOME>/op_admin`. This requirement stems from Sentinel's overall file system organization. Because the OPNET directory and model directories can be placed anywhere in the file system, Sentinel programs must have a fixed location from which to obtain the locations of these directories. The administration directory serves as a fixed site for this configuration information relative to the current user's home directory.

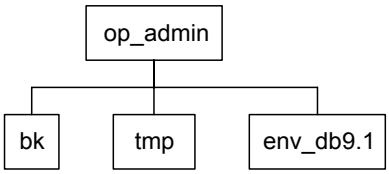
The initial organization of op_admin is established at installation. Two empty subdirectories and two files are also created within op_admin:

Table 2-6 Contents of the op_admin Directory

Subdirectory	Contents
bk	The backup directory stores user model backups, which are automatically created by the spsentinel program at user-specified intervals.
tmp	The temporary file directory stores any temporary files created by Sentinel programs.
env_db11.5	The environment database specifies user configuration data and preferences for Sentinel programs (including the above-mentioned directory location information).
End of Table 2-6	

The following figure shows the initial organization of the op_admin directory.

Figure 2-6 Initial Organization of Administration Directory



The numerical part of the environment database file name is the Sentinel release to which the database applies. The release number prevents conflicts when an earlier installation is still in use. Sentinel documentation sometimes refers to these files generically by omitting the release number (for example, env_db).

Backup Directory (bk)

Sentinel creates two types of backup files:

- time-based backup files preserve all model files being edited at the time the backup occurs. You can specify the backup interval (by default set at 15 minutes) with the backup_interval preference.
- version-based backup files are created each time you modify and save an existing model file. You can specify the number of versions saved with the backup_max_count preference.

Both types of backups are stored in the backup directory (`<HOME>/op_admin/bk`) with file names based on the name of the corresponding model.

Version-based backups are named following this format:

`<model_name>_bk.<n><file_type_suffix>`

— where <n> is a consecutive number that will not exceed backup_max_count.

Time-based backup files are named following this format:

`<model_name>_bk.<file_type_suffix>`

If a name has not been assigned to a model when a time-based backup occurs, it has the default name “unnamed”. This file overwrites any older unnamed model backups of that type.

The file suffix identifies the type of file. A list of these suffixes appears in Model File Naming Conventions on page SSR-2-18.

You can make a model backup a regular model file by moving or copying it into a model directory, using one of the following commands.

Sample UNIX command-line: `% mv net1_bk.nt.m net1.nt.m`

Sample Windows command-line: `C:\> move net1_bk.nt.m net1.nt.m`

The backup model then appears in the relevant model menus.

Temporary File Directory (tmp)

The temporary file directory (`<HOME>/op_admin/tmp`) contains temporary files generated by several Sentinel programs in the course of execution, such as the print output files generated by spsentinel printing operations. There are two locations where applications traditionally store such files—the current working directory and the system temporary directory (`tmp` in UNIX, and`%tmp%` or `C:\TEMP` in Windows). Both of these locations have drawbacks:

- The current directory is an unreliable location for storing temporary files because it sometimes will not allow their creation due to read-only security settings.
- Under UNIX, the tmp directory is located in the root (/) partition, which is commonly allocated minimal free space (typically less than 16 MB). This lack of space almost invariably leads to serious problems (such as crashed programs) when tmp fills up with temporary files.

Because of these problems, Sentinel programs do not use the current directory or the system temporary directory to store temporary files. Instead, they store these files in the Sentinel temporary file directory. This location has three advantages:

- The permissions for this directory are reliable, because it is created when Sentinel is installed.
- The amount of free space in the partition that contains this directory is usually larger than the amount of free space in the root partition.
- There is greater flexibility for removing unnecessary files.

Temporary File Types and Names

The Sentinel temporary file directory can contain the following types of files:

- Message files used for inter-process communication between the spsentinel program and executing simulations. The names of these files use the following formats:

  `ctp_<process_id>.<sequence_num>`

  `ptc_<process_id>.<sequence_num>`

  Message files, which contain text and special flags, normally exist in tmp only during simulation execution. However, if a simulation terminates abnormally, message files may be left behind. The message file names are based on the direction of data transfer: “ctp” (child-to-parent) files are used to send simulation information to spsentinel and “ptc” (parent-to-child) files are used to send spsentinel information to the simulation.

In the preceding name formats, <process_id> is the number of the process that generated the file and <sequence_num> is an integer representing the order in which the file was generated by the process, beginning with 0.

Note—Do not confuse the “.ps” suffix of PostScript files with the “.ps.c” and “.ps.o” suffixes of certain model files.

Temporary File Directory Maintenance

Most of the files stored in tmp have value for a limited time only, after which they can be removed. Typically, the only files worth keeping are captured graphics or PostScript files that must be reprinted periodically. You should move such files from tmp and store them elsewhere to avoid accidental removal.

During periods when a great deal of printing is being performed through spsentinel, the temporary directory tends to fill up with files. It is a good idea to periodically clear out tmp.

User Model Directories

The SP Sentinel models you develop are stored in one or more directories called model directories. The spsentinel program imposes no fixed location for model directories; you have complete flexibility in selecting where in the file system models should be stored. You indicate which directories contain models via a list stored in the mod_dirs preference (this mechanism is discussed in section mod_dirs on page SSR-3-24). However, there are several recommended conventions for model directories; these are initially set up during the installation procedure.

Because the mod_dirs preference is stored in the environment database (<HOME>/op_admin/env_db11.5), each user can specify a different set of model directories. Typically, you will take advantage of this by storing personal models in one or more model directories in your home directory. As a convenience, a model directory named op_models is created in your administration directory and includes this directory in the list of model directories. You can change this after installation, completely removing op_models if desired.

Model Directory Organization

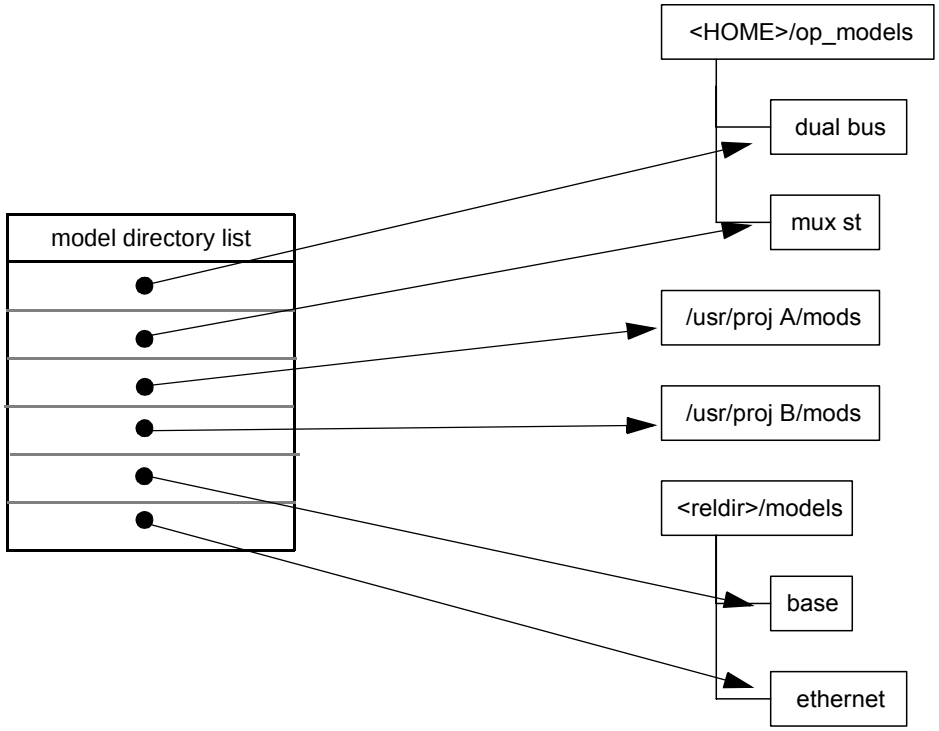
Most users want convenient access to the models provided with the Sentinel release media, both for reference and use. All of these models are stored in subdirectories of the standard models directory (<opnet_dir>/<reldir>/models/std). The most important of the standard models are the advanced models in the base subdirectory, which include fundamental and default models. Because most Sentinel users will rely on these models, the base subdirectory is included in the list of the model directories set up during installation. The other standard models, which include various applications such as LANs and mobile networks, are often useful and are also included in the list of default model directories.

The default list of model directories thus includes several directories:

- op_models in the home directory
- each of the subdirectories in the models directory

You can modify this simple configuration as more models are developed and a structure for organizing models is defined. An example of a complex configuration is shown in the following figure. It includes two subdirectories under models, several user model directories, and several shared model directories. This type of configuration is recommended once you become familiar with SP Sentinel and need enhanced organization of models.

Figure 2-7 Example of a Complex Model Directory Organization



Model Precedence

The order of directories in the mod_dirs list is important because it determines the precedence of model searches. For instance, if network models named mt_lan are stored in two different model directories, Sentinel programs will find the model stored in the directory listed earlier in the mod_dirs list and ignore the second model stored in a directory appearing later in the list. Thus, you should observe the following guidelines when listing model directories (and icon database directories, with are also listed in mod_dirs):

- Place the default directory for new models first in the mod_dirs list (as explained in the following, Default Model Directory on page SSR-2-18).
- Place more actively used model directories and directories with custom icon databases earlier in the list, so that their contents will take precedence in the event of name conflicts.
- Place subdirectories of the models directory near the end of the list, because any user models with duplicate names should take precedence over original models.
- Place the icon directory (icons) last in the list.

Default Model Directory

The first model directory in the mod_dirs list has a special significance and is termed the default model directory. This directory is the default location where new models are stored when written out or created by Sentinel programs. For instance, if you enter spsentinel, open a new project, define a new network model, and save it, it will be stored in the default model directory. If the default model directory is not the appropriate location for a new model, you can move it to the desired model directory via the UNIX mv or Windows move command. On the other hand, models that are not new (because they already exist within the model directories) are written out to the locations where they were stored when Sentinel began execution (or the last time a Rehash operation was invoked in Sentinel).

Model File Naming Conventions

Sentinel programs look for models in the model directories, which are listed in the mod_dirs preference. Within each model directory, models are stored as ASCII, binary data, or object code files, on a “one file per model” basis.

Versioning in File Names

Certain model files may be updated from one release to the next, so a version number (v<n>) is integrated into the file name, as shown in the examples below:

ams_aal1_conn_v3.pr.m

ams_support_v2.ex.c

Reference the version number to verify that you are using the most recent version of a particular model. Version numbers may be included in the following file types:

- process model files (.pro.m)
- header files (.h)
- packet format files (.pf.m)
- ici format files (.ic.m)
- external code files (.ex.c, .ex.h)

File Names for Compiled Code

Files that contain compiled code (such as object files) are specific to a particular kernel or machine architecture. Thus, the names of such files contain suffixes to characterize the code contained in the file.

Code Type Suffix The file names for object files, repositories, and simulation executables include a suffix that identifies what kind of code the file contains. This suffix is used to prevent different compilation modes from overwriting one another's files. The code type suffix (listed in Table 2-7) appears before the architecture suffix, as in `sink.mtdev32.s1.o`.

Table 2-7 File Name Suffixes for Code Type

Suffix	Type of Code
dev32	32-bit integer, debugging mode, sequential
mtdev32	32-bit integer, debugging mode, parallel
opt32	32-bit integer, optimized mode, sequential
mtopt32	32-bit integer, optimized mode, parallel
dev64	64-bit integer, debugging mode, sequential
mtdev64	64-bit integer, debugging mode, parallel
opt64	64-bit integer, optimized mode, sequential
mtopt64	64-bit integer, optimized mode, parallel
End of Table 2-7	

Architecture Suffix All OPNET model files are portable across different machine architectures. However, simulation executables (`.sim`), object code files (`.<md>.o`), and simulation archives (`.si.a`) are specific to a given machine architecture.

Architectures are characterized by CPU type (which determines the object code format) and OS type (which determines the variations of library functions). The architecture suffix incorporates these data items in a two-character code, as follows:

```
<CPU_char><OS_number>
```

The following table lists the architecture codes used by OPNET.

Table 2-8 Architecture Codes

Code	CPU Type	Operating System	Vendor
i0	Intel x86	Windows	Microsoft
s1	SPARC	Solaris	Sun

The architecture suffix is optional. It's use is controlled by a preference, described in arch_suffix on page SSR-3-17. The following examples show the names of OPNET files created with arch_suffix set to both FALSE and TRUE.

arch_suffix = FALSE

```
eth_mac.dev32.pr.obj
eth_net.opt32.sim
```

arch_suffix = TRUE

```
eth_mac.dev32.i0.pr.obj
eth_net.opt32.s1.sim
```

File Type Suffixes

Sentinel programs determine the type of each model file from a unique file name suffix associated with each type. This mechanism is common in the UNIX and Windows environments, where files are suffixed with a code for the programming language used (for example, ".c" for C files, ".f" for FORTRAN, and ".o" for object code). UNIX and Windows suffix conventions are integrated within the Sentinel conventions.

The following table lists Sentinel file suffixes.

Note—Object files end in `.obj` on Windows and `.o` on UNIX. Libraries end in `.dll` on Windows and `.so` on UNIX.

Table 2-9 SP Sentinel File Type Suffixes (Part 1 of 6)

File Type Suffix	Description	Format
.ac	Analysis Configuration	binary data
.ace.f	ACE Filter Definition	ASCII text
.ace.hm	ACE Host Mapping (obsolete)	ASCII text
.ace.i	ACE Import Definition (obsolete)	ASCII text
.ace.ic.txt	ACE Import Configuration Files	ASCII text
.ace.mt.txt	ACE Multi-Transaction Report Settings	ASCII text
.ace.qpb	ACE QuickPredict Bar Chart Templates	ASCII text
.ace.qps	ACE QuickPredict Graph Templates	ASCII text
.ace.rt.txt	ACE MS Word Report Settings	ASCII text
.ace_dict	ACE Protocol Dictionary	ASCII text
.ad.m	Public Attribute Description	binary data
.af	Automation Settings File	binary data

Table 2-9 SP Sentinel File Type Suffixes (Part 2 of 6)

File Type Suffix	Description	Format
.agents	ACE Agents files	ASCII text
.ah	Animation History	binary data
.alias	Alias Modifications	ASCII text
.as	Animation Script	ASCII text
.atc.m	ACE Application Task Characterization	binary data
.bkg.i	Background map Image	binary data
.bmp	BMP Image	binary data
.cds	Cartographic Data Set	binary data
.cml	Custom Model List	ASCII text
.csv	Comma Separated Values Data File	ASCII text
.demand.d	Derived Demand Model	binary data
.demand.m	Demand Model	binary data
.dict	TIF Dictionary	binary data
.edge	Edge Devices Information	ASCII text
.ef	Environment File	ASCII text
.el	Iso-Line/Cartographic Data Set	binary data
.em.x	EMA Application	executable form
.esd.m	External System Definition Model	binary data
.ets	External Tool Support File	ASCII text
.ets.so	External Tool Support	shared library
.fl.m	Filter Model	binary data (editable form)
.fl.x	Filter Model	binary data (executable form)
.gdf	General Purpose Data File	ASCII text
.gif	GIF Image	binary data
.grf	Grouping Rules File	ASCII text
.hcon.l	Hardware Configuration Library	ASCII text

Table 2-9 SP Sentinel File Type Suffixes (Part 3 of 6)

File Type Suffix	Description	Format
.hlp	Help File	ASCII text
.html	HTML File	ASCII text
.ia	Import Assistance File	ASCII text
.ic.m	ICI Format	binary data
.icons	Icon Database	binary data
.image	Generic Image	binary data
.imp.log	Import Log	ASCII text
.lk.d	Derived Link Model	binary data
.lk.m	Link Model	binary data
.m.conv	Model Conversion Script	
.ma	Model Assistance File	ASCII text
.map.i	Image Map	binary data
.mce.m	Mainframe Workloads	binary data
.md.m	Modulation	binary data
.mid	MapInfo Data Interchange Format File	ASCII text
.mif	MapInfo Data Interchange Format File	ASCII text
.nd.d	Derived Node Model	binary data
.nd.m	Node Model	binary data
.nds	Network Diff Selection	ASCII text
.nt.log	Simulation Log	ASCII text
.nt.m	Network Model	binary data
.nt.so/.nt.dll	Network Repository	shared library
.of	Output Fragments	binary data
.orb	Satellite Node Orbit	binary data
.os	Output Scalars	binary data
.ot	Output Tables	binary data

Table 2-9 SP Sentinel File Type Suffixes (Part 4 of 6)

File Type Suffix	Description	Format
.otf	Fragmented Output Tables	binary data
.ov	Output Vectors	binary data
.ovd	Output Vector Dictionary	ASCII text
.pa.m	Antenna Pattern	binary data
.path.d	Derived path object	binary data
.path.m	Path object	binary data
.pb.m	Probe List	binary data
.pbr.m	Report Probe Model	binary data
.pbs.m	SLA Probe Model	binary data
.pd.m	Probability Density Function (PDF)	binary data (editable form)
.pd.s	Probability Density Function (PDF)	binary data (simulation loadable form)
.pdf	Portable Document Format (Adobe Acrobat file)	binary data
.pf.m	Packet Format (obsolete)	binary data
.pk.m	Packet Format	binary data
.ppl.m	Public Profile Model	binary data
.pps.m	Private Profile Storage	binary data
.pr.m	Process Model	binary data form
.prj	Project Model	object code form
.prop.d	Propagation Model	binary data
.prop.p	Propagation Parameter Set	binary data
.prop.r.so	TMM Propagation Model	shared library
.py	ETS Code	Python source code
.pyc	ETS Code	Python byte code
.repos	Repository	shared library
.repos.so/.repos.dll	Simulation Repository	shared library
.rspdb	Response database	binary data

Table 2-9 SP Sentinel File Type Suffixes (Part 5 of 6)

File Type Suffix	Description	Format
.sce.f	Server Workload Filter Library	binary data
.sce.m	Server Workloads	binary data
.sce.t	Server Workload Template	binary data
.sce.x	Server Workload Export	XML
.scf	Satellite Configuration	binary data
.sched	Task Scheduler data (Automation module)	binary data
.sched.log	Scheduled Tasks Log	ASCII text
.sd	Simulation Description	ASCII text
.sddb	Self Description database	binary data
.sel.f	Logical Object Selection Filter	ASCII text
.selset	Selection Set file	
.seq	Simulation Sequence	binary data
.sim	Simulation Executable	executable program
.supr.xml	NetDoctor Suppressions	XML
.tim	Traffic Import Model	object code
.trj	Mobile Node Trajectory	ASCII text
.urep.xml	User Report Table	ASCII text
.val.r	NetDoctor Rule (Pre-10.0)	ASCII text
.val.rpt	NetDoctor Report Template (Pre-10.5)	ASCII text
.val.rr	NetDoctor Rule	ASCII text
.val.xrpt	NetDoctor Report Template	XML
.vd.m	OPNET Vector Drawing	binary data
.vdf	View Definition File	ASCII text
.wdomain.d	Derived Wireless Domain	binary data
.wdomain.m	Wireless Domain	binary data

Table 2-9 SP Sentinel File Type Suffixes (Part 6 of 6)

File Type Suffix	Description	Format
.wrc	UI Configuration Resource	binary data
.xml	XML File	XML
.xsl	XSL file	XML
End of Table 2-9		

Customizing the User Interface

You can make the following changes to the Sentinel user interface:

- Add shortcuts for menu items or change existing ones
- Add optional operations

You accomplish these changes by editing the files that control the interface. These files have the file extension ".ets" (external tool support).

Editing Keyboard Shortcuts

Shortcuts are key combinations that substitute for selecting a particular menu item. For example, pressing F1 has the same effect as choosing What's This? from the Help menu. You can change the defined shortcuts or add new ones.

Procedure 2-1 Adding or Changing a Shortcut

1 From the system prompt, move to the `<reldir>/sys/configs` directory.

2 Locate the .ets file that contains the operation definition of the operation you will create a shortcut for. Check each of the following files in turn, until you find the desired operation definition:

- std_itguru_project_operations.ets
- std_help_operations.ets
- std_edit_operations.ets
- std_shortcut_operations.ets

➥ Each of these files consists of numerous operation definitions, as shown in the following example.

```
# button list for Project Editor
File format: operation list

start_operation
   menu header:   "File"
   menu string:   "Close Project"
   operation:     exec_builtin close_project
   position:      2
   shortcut:      "ctrl-w"
   separator      "yes"
end_operation

start_operation
   menu header:   "File"
   menu string:   "Save Project"
   operation:     exec_builtin save_project
   position:      4
   shortcut:      "ctrl-s"
end_operation
```

3 Use the text editor's search capability to locate the operation you want to modify. The text string shown as "menu string" duplicates the menu item shown in Sentinel.

4 Change the shortcut defined, or add a line for a new shortcut. Shortcut keys can be function keys (<F1>, <F2>) or control key combinations (specified as ctrl-<key> in the operation definition).

5 Save the changes to the file, close the text editor, and start Sentinel.

6 Test the new shortcut. Ensure that:

- It appears on the menu list next to the operation.
- It has the functionality intended.
- It appears in the Shortcuts list. This list is displayed when you choose Shortcuts from the Help menu.

End of Procedure 2-1

Adding Optional Operations

Sentinel includes code for some optional operations that are not normally included in the menus, such as the Edit Comment Log operation. You can add action buttons or menu items for these operations.

Procedure 2-2 Adding an Optional Operation

1 From a system prompt, move to the `<reldir>/sys/configs` directory.

2 In a text editor, open the `std_itguru_project_operations.ets` file.

3 Add operation definitions for the optional operations to the file. The lines shown below create the following effects:

- User lock and comment log operations are added to the File menu. User lock is the last item in the list; comment log is next to last. The position of the menu item in the list is controlled by the position variable.
- Buttons for both operations appear in the button bar. The position of the button in the button bar is determined by the order of the operation definition in the file (for example, the first operation listed will be the left-most button).

```
start_operation
   menu header:   “File”
   menu string:   “Comment Log”
   operation:     exec_builtin open_comment_log
   position:      9998
   icon:          comments
end_operation


start_operation
   menu header:   “File”
   menu string:   “User Lock”
   operation:     exec_builtin toggle_user_lock
   position:      9999
   icon:          lock_unlock
end_operation
```

4 Save the changes to the file, close the text editor, and start Sentinel.

5 Test the new operation. Ensure that:

- It appears on the menu list in the desired position.
- It appears in the button bar in the desired position.
- It has the functionality intended.

End of Procedure 2-2

3 Preferences

Preferences provide a uniform system for passing options and parameters to Sentinel programs. In most cases, you can set the values of preferences from the Preferences menu item on the Edit menu, as described in Preferences on page SSU-4-36 of the *User Guide*. This section describes how preferences work and gives other methods of setting their values. The later sections describe the common preferences.

Scope

Preferences can be common or program-specific in scope. Common preferences are supported by most or all Sentinel programs. These preferences are grouped in sets based on their use. The most general of these sets is described in Standard Preferences on page SSR-3-16 The remaining sets are described in Preference Sets on page SSR-3-29.

Program-specific preferences apply to one program and provide information needed by that program alone. These preferences are described with each program in the Program Descriptions chapter.

You can use any common preference as a program-specific preference by adding a program-name prefix, as follows

```
<program_name>.<attribute_name>
```

You can use both the common and program-specific versions of a preference together. For example, your environment database might include the assignment:

```
diag_enable: false
```

This assignment would apply to all Sentinel programs. You could then override this assignment for one run and program by specifying the program-specific version as a command line flag:

```
<prog_name> -<prog_name>.diag_enable true
```

As implied by the preceding example, Sentinel gives precedence to program-specific preferences over the corresponding common preferences.

Data Types

There are several types of preferences, some of which are more common than others. The four basic types are:

- integer
- double (double-precision real number)
- character string
- boolean (with possible values of TRUE and FALSE)

Other types are:

- password
- lists made from the four basic types:
  - integer list
  - double list
  - string list
  - enum (a string list with a defined set of values)
  - boolean list

A list is an ordered set of zero or more items of the appropriate type. Each of these data types has a specific syntax associated with it for performing an assignment. The syntax varies depending on the assignment mechanism.

Passwords

The password data type is used for preferences that contain a password. For security, Sentinel automatically encrypts password values before storing them in the environment database. It is difficult (but not impossible) to decrypt the value of a password preference.

Encrypted passwords appear in the Preferences dialog box and in the environment database as a case-sensitive string of characters preceded by “op_password:” and, possibly, terminated by “=” characters. For example:

```
op_password:X8GU00l72t5eX6uywantkQ==
```

If you enter a clear text password directly into the environment database, Sentinel automatically encrypts the password the next time it reads the env_db file (at program start up).

Note—Clear text passwords must never begin with the string “op_password:”.

Assignment Mechanisms

There are several mechanisms for assigning preference values from the environment of a program. Consistency among the different mechanisms is kept as close as possible, given that the mechanisms must abide by traditions common to the host operating systems. The goal is for any option or parameter of a program to be assignable using any of these mechanisms (in descending order of precedence):

- A custom environment (`.ef`) file (in one of the mod_dirs directories)
- The command line
- Shell variables
- The default environment database (env_db)
- A product environment file (`<product>.ef`). Product environment files are located in `<reldir>\sys\configs\global_prefs`
- The Windows registry

Command line flags and shell variables are universally used mechanisms for utility programs; environment files and environment databases are Sentinel-specific, but similar to the preference files used in other systems (such as the X Window System). Each mechanism is described separately below.

Command Line Flags

Most UNIX and some Windows programs accept command line flags. It is standard to precede command line flags with a dash. The UNIX `ls` directory-listing program is an example: the `-l` flag indicates that it should print an extended listing of the files in the current directory:

```
ls -l
```

Sentinel programs also abide by the dash convention. Each preference that a program accepts can be assigned a value via a command line flag composed of the preference name preceded by a dash and followed by its value. The following program invocation example has three flags, which set three different preferences.

```
op_vuorb -orb_name leo -sm_axis 6700.0 -moon_effects
```

Most preferences take a value argument: in the preceding example, the sm_axis preference is assigned the value 6700.0 and the orb_name preference is assigned the value “leo”. On the other hand, the boolean moon_effects preference does not need a value argument; merely including it on the command line is equivalent to setting its value to TRUE.

The command line syntax for each of the four basic preference types is shown below:

Table 3-1 Command Line Syntax for Basic Preference Types

Type	Syntax	Example
integer	-<attr_name> <int_value>	-size 345
double	-<attr_name> <double_value>	-mean_anomaly 214.65
string	-<attr_name> <string_value>	-orb_name statkeep -site_name "VLA"
boolean	-<attr_name> -<attr_name> <boolean_value>	-verbose -verbose OFF
End of Table 3-1		

where <*boolean_value*> is defined to be one of the following strings: On/Off, True/False, and Enabled/Disabled. These boolean values are not case sensitive. Any combination of upper- and lower-case letters is valid.

The following syntax considerations apply to preferences used as command line flags.

- Strings that are one word (with no embedded spaces or tabs) can appear directly after the flag name. Strings consisting of multiple words must be delimited by double quotes.
- Strings that contain a dash (such as negative numbers and flags being passed to another program) must be specified in one of the following ways:
  - — Enclose the value in double quotes, with a space character before the dash. For example: `-bind_flags " -g"`
  - — Change the dash to a caret (which will be converted back to a hyphen automatically). For example: `-bind_flags ^g`
- Boolean flags have two forms:
  - — A short form in which no value argument is supplied (the value TRUE is assumed).
  - — A long form in which a boolean value argument is supplied.
- List preferences use an extension of basic preference syntax. Instead of one value argument following each preference name flag, there can be more than one value. An example of a string list preference being assigned via the command line appears below. In this case, the ef preference is assigned two strings: “base_orb_defs” and “low_earth_defs”.

```
op_vuorb -ef base_orb_defs low_earth_defs
```

The command line flag mechanism for preference passing is useful for preferences that change on a run-to-run basis (for instance, the “verbose” preference). However, it is not very convenient for large numbers of preferences. For programs that require a large number of preferences, it is more efficient to pass most of the preferences via the environment file mechanism described under Environment Files on page SSR-3-9.

Shell Variables

Shell variables are an alternative way to pass preferences to Sentinel programs. However, they are less convenient than the other mechanisms and are less likely to be used. Shell variables provide a good mechanism for setting a preference temporarily (that is, for the lifetime of the current command interpreter) that is to be passed to a number of programs (or to many invocations of the same program).

The following tables show assignment syntax for shell variables of the four basic preference types. The syntax in the first table applies to the C shell (`/bin/csh`) available in all supported versions of UNIX. The alternate Korn shell (`/bin/ksh`), Bourne shell (`/bin/sh`), and T shell (`/bin/tsh`) have a slightly different syntax for shell variable assignments. The second table lists Windows command interpreter (`cmd.exe`) syntax.

Table 3-2 UNIX Shell Variable Syntax for Basic Preference Types

Type	Syntax	Example
integer	setenv <attr_name> <int_value>	setenv size 345
double	setenv <attr_name> <double_value>	setenv mean_anomaly 214.65
string	setenv <attr_name> <string_value>	setenv orb_name statkeep setenv site_name "VLA"
boolean	setenv <attr_name> <boolean_value>	setenv verbose OFF setenv verbose TRUE
End of Table 3-2		

Note—The short form of boolean preferences permitted by command line flags (including only the flag without a subsequent value argument) is not supported by shell variables.

Table 3-3 Windows Shell Variable Syntax for Basic Preference Types

Type	Syntax	Example
integer	set <attr_name>=<int_value>	set size=345
double	set <attr_name>=<double_value>	set mean_anomaly=214.65
string	set <attr_name>=<string_value>	set orb_name=statkeep set site_name=VLA
boolean	set <attr_name>=<boolean_value>	set verbose=OFF set verbose=TRUE
End of Table 3-3		

Note—Do not use quotes when setting string values in Windows, even if there are several space-separated words in the value.

The syntax for assigning a sequence of values to list preferences using shell variables requires delimiters to separate individual values; the delimiter character is the colon (":"). An example of a string list preference being assigned via a shell variable appears below. In this case, the ef preference is assigned three strings ("base_orb_defs", "low_earth_defs", and "options").

```
setenv ef "base_orb_defs:low_earth_defs:options"
```

Under some circumstances, the automatic parsing of shell variables by Sentinel programs may be undesirable. For instance, a shell variable assignment that is not related to Sentinel may conflict with a preference with the same name; this might override the correct assignment from the environment database. This is most likely to occur for preferences with short names, such as "sys". In such cases, the conflict can be resolved at the expense of losing access to all shell variables for preference assignments. To disable parsing of shell variables by the environment package, set the special boolean preference no_shell to TRUE (this preference will have its intended effect only if assigned within the environment database, as shown).

```
no_shell: TRUE
```

Environment Database

The environment database serves as a "preference file" for Sentinel programs. It is a technique by which preferences can be consistently assigned desired values with no special work to perform when a program is invoked, because the environment database is automatically loaded and scanned by all Sentinel programs. The environment database has the same syntax as a normal environment file; it just has a special name (`env_db`11.5) and a special location (`<HOME>/op_admin`) that identifies it as the environment database. You can change the contents of the environment database by editing it directly or via the Preferences menu item on the Edit menu.

An example environment database appears in the following figure.

Figure 3-1 Environment Database Example

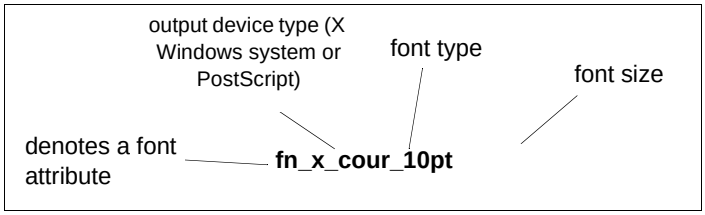
```
# Environment Database


opnet_dir:    /usr/opnet
mod_dirs:    /usr/users/tsmith/op_models,
/usr/opnet/11.5.A/models/std/base,
            /usr/opnet/11.5.A/sys/icons


verbose:     FALSE
vudoc_prog:  acroread


fn_x_cour_8pt:     Courier8
fn_x_cour_10pt:    Courier10
fn_x_cour_12pt:    Courier12
fn_x_cour_14pt:    Courier14
fn_x_times_18pt:   Times-Roman18
fn_x_helv_14pt:    Helvetica12
```

The opnet_dir and mod_dirs preferences are defined because they are required by most Sentinel programs. The vudoc_prog and verbose preferences are set to typical values. The other preferences being set are font mappings, which map Sentinel font names to specific output device fonts available on the display. For example, the Sentinel font fn_x_cour_10pt maps to the X Windows System Courier 10-point font. The naming scheme for Sentinel fonts is described below:



The output device types can be set to x (X Windows system) or ps (PostScript); font types include Courier, Times, and Helvetica; and font sizes range form 8 to 18 points. Any of the default fonts can be mapped to a different system font (e.g. fn_x_cour_12pt could be set to Times-Roman 10), but the preference names cannot be changed. You may also notice several fonts in the environment database which appear in the form fn_x_annot_<*size*>. These fonts are used by annotation text and can be set to either small (Courier 10), medium (Courier 12), or large (Courier 14) in the text preferences. However, these default font types and sizes can be changed in the environment database to any system font.

Font preferences are used by GUI-based programs, but do not appear in the documentation of these programs. This is because these preferences are dynamic preferences that are searched for in the environment at the point in program execution when their value is needed. In contrast, static preferences (those documented in the Program Descriptions chapter) are searched for in the environment shortly after a program commences execution. Font requests have been implemented as dynamic preferences to allow enhanced flexibility and efficiency; the five font preferences shown in the preceding example are not the only dynamic preferences used by Sentinel programs. Dynamic preferences are discussed further in Static vs. Dynamic Preferences on page SSR-3-12

Environment Files

The environment file mechanism is a way of bundling together many preference assignments and passing them to a program as a block. Environment files are ASCII text files with a simple syntax for making assignments. Generally, each line of an environment file constitutes an assignment to one preference. The assignment syntax for environment files appears in the following table.

Table 3-4 Environment File Syntax for Basic Preference Types

Type	Syntax	Example
integer	<attr_name>: <int_value>	size: 345
double	<attr_name>: <double_value>	mean_anomaly: 214.65
string	<attr_name>: <string_value>	orb_name: statkeep site_name: 'VLA'
boolean	<attr_name>: <boolean_value>	verbose: OFF verbose: TRUE
End of Table 3-4		

In environment files, any amount of white space can occur between the preference name, the colon separator, and the preference value. In fact, assignments can spread out onto multiple lines. Comments can also appear in the file, using the shell script standard of starting a comment with a pound sign (“#” anywhere on a line, with or without assignments).

Figure 3-2 Environment File Example

```
orb_name:       low_earth
elem_type:      osc

sm_axis:        6700.0
eccen:          0.02
inclin:         30

start_date:     1989.01.01
start_time:     0.0.0.0
stop_date:      1989.01.01
stop_time:      12.0.0.0

time_step:      60.0

moon_effects:   on
sun_effects:    on
srp_effects:    off     # this is off for speed
drag_effects:   on
```

The syntax for assigning a sequence of values to list preferences using an environment file requires delimiters to separate individual values; the delimiter character is the comma (“,”). An example of a string list preference being assigned via an environment file appears below. In this case, the ef preference is assigned three strings (base_orb_defs, low_earth_defs, and options)

```
ef: base_orb_defs, low_earth_defs, options
```

Preference names that contain embedded spaces must be enclosed in quotes when used in an environment file

```
"HLA Federate Name" : OPNET
```

You can create environment files with any text editor. The file names must have a ".ef" suffix to be recognized as environment files:

```
polar.ef
equatorial.ef
```

To be accessible to Sentinel programs, environment files must be placed in one of the model directories referenced by the mod_dirs preference. Environment file names must be passed to a program through the command line flag `-ef`. Do not include the environment file suffix in the file name argument:

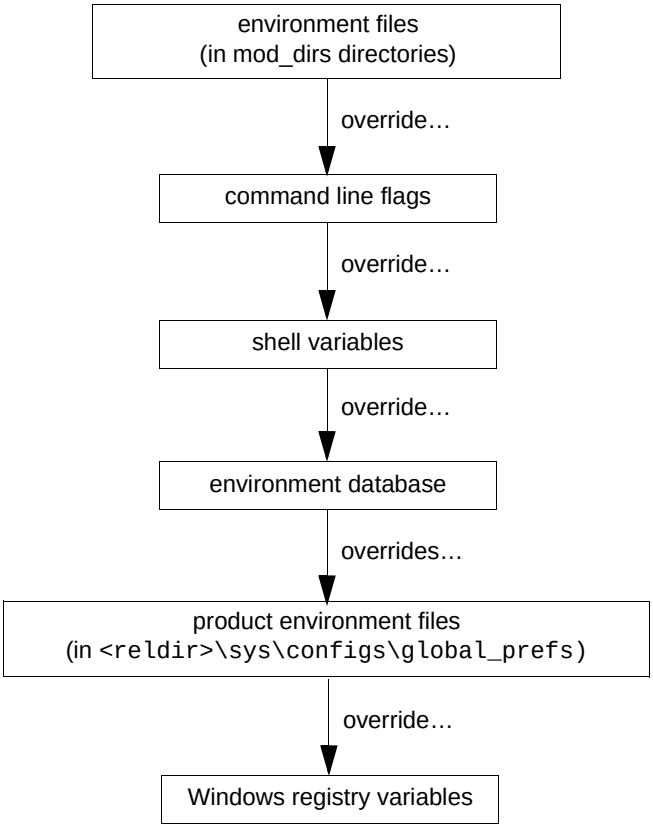
```
op_vuorb -ef polar
```

Note—A product-specific environment file (spsentinel`.ef`) is located in the `<reldir>/sys/configs/global_prefs` directory. Typically, you won't need to change this environment file.

Assignment Precedence

With five different mechanisms for passing preferences, there is a possibility that the same preference will be assigned via more than one mechanism. Figure 3-3 shows the order of precedence for preference passing mechanisms. This order is fixed and unmodifiable. Preference assignments that are overridden by higher precedence mechanisms have no effect.

Figure 3-3 Assignment Precedence for Preferences



Within a given mechanism, there is also a possibility of conflicting preference assignments. The rules of precedence for each mechanism are similar: later assignments of a preference override earlier assignments. Thus, for environment files (and the environment database), assignments that appear later in a file will override conflicting definitions that appear earlier in the file. Likewise, command line flags that appear later on the command line will override conflicting flags on the same line. Shell variable assignments are similar: later assignments overwrite those made earlier.

An exception to the precedence rule is that an assignment of the form <program name>.<preference> overrides assignments of the form <preference>, even if the latter is from a higher precedence source. For example, if spsentinel`.icons` is defined in a shell variable but `icons` is defined in an environment file, OPNET uses spsentinel`.icons` as the value of its icons preference, even though `icons` is assigned in a higher-precedence source (an environment file).

Static vs. Dynamic Preferences

There are two types of preferences that can be assigned: static and dynamic. *Static preferences* form the fixed set of options that is built into the program as a part of its interface. *Dynamic preferences* hold values for program options that the program determines it needs while executing. Dynamic preferences differ from static ones in that they can be based on conditional events in the execution of a program and thus cannot be considered part of the built-in, centralized option set of a program.

Static preferences have the following characteristics:

- They are searched for in the environment shortly after a program begins executing.
- They can be required or optional. A *required static preference* halts execution of the program if you do not assign it a value. An *optional static preference* has a default value assigned to it automatically by the program if you do not assign a value.
- They are displayed in the list produced by the “help” preference (discussed in help on page SSR-3-22).
- They are described in the Program Descriptions chapter.

Dynamic preferences have the following characteristics:

- They are searched for in the environment whenever an executing Sentinel program determines a need for a preference value.
- They can be entered in response to a prompt. When a dynamic preference is requested by Sentinel and an assignment for it is not found in the program’s environment, a prompt appears on the standard output device requesting you to enter a value for the preference. You can either enter a value or press the <Return> key to assign a default value. (This constitutes a fifth mechanism for assigning dynamic preference values; it is not available for static preferences.)

- Their use is limited to the following types of preferences:
  - — Font name mappings of GUI-based programs.
  - — The arch_suffix preference, which indicates whether a two-character abbreviation of the host architecture (CPU type and OS) should be added to the file type suffix of object-code model files.
  - — Dynamic requests for simulation preferences.
  - — Patch preferences (such as xpatch_polyxor_vendors and xpatch_blitclip_vendors) that indicate deficiencies in the host environment.
  - — Compatibility preferences (such as outfile and logfile) from previous releases of Sentinel.
- They are not described in the Program Descriptions chapter, due to their dynamic and conditional nature.

Name Wildcards

Sentinel simulations use dynamic preferences to obtain values for promoted object attributes. In some cases, the number of dynamic preferences used by a simulation can be large. This usually occurs when many nodes are involved, each with promoted attributes. Manually entering values for each dynamic preference would be tedious in such cases, especially if many of the preference values were the same. To simplify the job of assigning multiple related preferences at the same time, Sentinel programs support a name wildcard feature.

Simulation object dynamic preferences have a hierarchical naming structure in which preference names are divided into a series of dot-separated components, each representing one level of the model hierarchy. Every name component belongs to one of the following categories:

- subnet name (one or more)
- node name
- attribute name

The format of hierarchical object attribute names is always top-down, but the number of name components can vary according to the type of object being identified. For instance, a promoted node attribute would be specified simply as `<node_name>.<attr_name>`. On the other hand, an attribute associated with several nested subnets would have a much longer hierarchical name, such as `top.<subnet1_name>.<subnet2_name>.<node_name>.<attr_name>`.

The wildcard feature allows the wildcard character (the asterisk, “*”) to match any number of alphanumeric characters (not including punctuation and special characters) in dynamic preference names. Thus, parts of an attribute name can be separately wildcarded, as follows.

```
The wildcard attribute name:

    *.*.tx[4].data rate

matches any of these actual attribute names:

    lan0.nd1.tx[4].data rate
    lan0.nd2.tx[4].data rate
    lan1.nd1.tx[4].data rate
```

The following wildcarding example shows an environment file statement that makes a wildcard assignment to all channels in a network model. Notice that the third wildcard character matches only the initial characters of the third part of the name; wildcard matching stops when it encounters a non-alphanumeric character (in this case, “[“).

```
*.*.*[*].data rate: 9600
```

The assignment syntax used for wildcarded dynamic preference names is generally the same as that used for attribute names without wildcards. However, command line assignments of wildcarded attribute names must have special protective symbols (double quotes) to prevent the command interpreter from interpreting the wildcard character as a wildcard for file names. In addition, the first character of a preference flag must be a dash (otherwise the flag will not be recognized as a flag). The proper syntax for a command line wildcarded attribute assignment is shown below.

```
net.sim "-*.*.qx.priority" 20
```

Notice that the double quotes enclose the whole sequence of characters as the first argument to `net.sim`, as well as protecting the special meaning of the asterisk character. The attribute’s value is not placed inside the quotes because it is a separate argument.

Runtime Prompts

If an assignment for a dynamic preference cannot be found anywhere in the environment, the program displays a prompt on the standard output requesting a value. This feature is especially useful with the runtime parameterization capability of simulations, which automatically promotes unassigned object attributes to dynamic preferences as they are encountered. The following example shows a typical runtime prompt.

```
---------------- Unset Environment Attribute -----------------
* lan0.e0.bus_tx[0].data rate: <double>
* Desc:
* Press <return> to select default: 10,000,000
Enter value> 100000000
--------------------------------------------------------------
```

To set a particular value for a prompted preference, type a value and press the <Return> key. By pressing <Return> without entering a value, you force the program to assign a default value to the preference. By typing the special keyword `noprompt`, you can prevent further prompting for dynamic preference values.

A dynamic preference may expect a list value. If so, you can enter multiple values on the same line, separated by commas. Terminate the list value entry by pressing the <Return> key.

```
---------------- Unset Environment Attribute -----------------

* top.m1.src.interarrival args: <arg. list>
* Desc:
* Press <return> to select default: 1.0

Enter value> 2.3, 4.818

--------------------------------------------------------------
```

Standard Preferences

There are several preferences supported by all Sentinel programs. These *standard preferences* provide common services that are useful to most programs. In general, they manage or analyze the program environment. Many of the standard preferences can be set only using a command line flag. These preferences are not supported by Sentinel scripts or batch files. The following table summarizes the standard preferences. Detailed definitions of each preference follow the table.

Table 3-5 Standard Preferences (Part 1 of 2)

Name	Description
arch_suffix	Specifies that an architectural suffix be included as part of the name of compiled files.
console_exit_pause	Specifies that the program displays a user prompt before terminating.
dconfirm[1]	Prints a list of all dynamic preferences used by a program.
ef[1]	Specifies a list of environment files to be loaded by the program.
env_db[1]	Specifies the full path name of the environment database to be used.
enva[1]	Lists (in alphabetic order) the environment assignments that are accessible to the program.
envp[1]	Lists (in precedence order) the environment assignments that are accessible to the program.
geocentric_model	Specifies which geocentric coordinate system to use for the earth.
handle_exception	Specifies whether the program (instead of the operating system) should handle memory access and other exceptions.
handle_exception_extended	Specifies whether to activate extended exception handling.
help[1]	Generates a list of all preferences the program can accept.
mem_clear[1]	Specifies that the program should clear *n* megabytes of memory before starting.
mem_optimize[1]	Specifies whether optimized memory-allocation routines should be used.
mod_dirs	Specifies the full path names of the model directories to be used.
new_env_db[1]	Causes the program to create a new environment database before running.
no_env_db[1]	Causes the program to ignore the user account's environment database (env_db11.5).
noprompt[1]	Causes the program to suppress prompts for the values of any unassigned dynamic preferences.

Table 3-5 Standard Preferences (Part 2 of 2)

Name	Description
opnet_dir	Specifies the full path name of the root directory for Sentinel.
opnet_user_home	Specifies the path name of the user home directory.
sconfirm[1]	Prints a list of all preferences and their values.
End of Table 3-5	

1. Can be assigned from the command line only.

arch_suffix

When TRUE, the arch_suffix specifies that an architectural code be included as part of the file name of compiled files. In a multi-architecture environment, this prevents files generated for one architecture from overwriting those for another architecture.

Data Type	boolean
Default Value	TRUE

Note—Setting the value of arch_suffix to FALSE causes Sentinel programs to ignore the affected files that have architectural suffixes. The files provided with Sentinel include architectural suffixes in their file names. Thus, if you choose not to use the architectural suffix capability, you must rename all affected files in the `<reldir>/models` subdirectories and any user model directories for the supplied files to be recognized by Sentinel programs.

console_exit_pause

When TRUE, specifies that the program sends a user prompt to the command line before terminating, and ensures that the console window remains open until you press <Return> or <Enter>. This allows you to view error and other messages sent to standard output, even if the program terminates abruptly.

To use this feature, make sure that both console_exit_pause and the console preference are set to TRUE (console specifies that error and output messages are sent to the application console).

Data Type	boolean
Default Value	FALSE

dconfirm

Prints information about each dynamic preference used by the program. The information printed is the preference's name, the source of the preference's current value, and the current value itself. The dconfirm preference can be used only from the command line.

Data Type	boolean
Default Value	FALSE

As with static environment variables, it is sometimes useful to confirm that dynamic preferences are being set to appropriate values. The dconfirm preference prints the value of each dynamic preference on the standard output as each value is requested by the program. It does not otherwise affect execution of the program.

The sconfirm preference is often used in conjunction with the envp and enva standard preferences, which print the environment settings in precedence and alphabetic order, respectively.

Figure 3-4 Sample dconfirm Preference Information

```
spsentinel-dconfirm

------------ Dynamic Environment Attribute Confirmation --------------

Name:                                  Source:        Value:
-------------------------------------------------------------------------
arch_suffix:                          default         TRUE
-------------------------------------------------------------------------
```

•
•
•

ef

Specifies a list of environment files to be loaded by the program. The order in which environment files are listed determines the order in which they are loaded and the precedence used if preference assignments conflict. Preference assignments made by files appearing later in the list will overwrite (that is, have higher precedence) than conflicting assignments made by files appearing earlier in the list. If no file name is specified, the network name will be used.

This preference can be used only from the command line.

Data Type	string list
Default Value	<empty>

The preference support built into all Sentinel programs allows any number of environment files to be loaded by a program, as specified by the ef preference. The preference assignments in these files supplement those taken from the environment database, shell variables, or command line flags.

The ef preference is generally assigned as a command line flag. An example assignment is shown below. Notice that the `.ef` suffix of the environment file name is omitted in the preference assignment.

```
net.sim -ef core_defs misc_defs
```

Note—Although the environment database (env_db11.5) is an environment file, it is automatically loaded due to its special role and does not need to be added to the ef list.

---

env_db

Specifies the name of an environment file to be used in place of the default environment database (env_db11.5).

This preference can be set from the command line, or as a shell variable. If this preference is set from the command line then it must be the first argument of the Sentinel program (unless new_env_db is also used).

Data Type	string
Default Value	“”

enva

Lists (in alphabetic order) the environment assignments accessible to the Sentinel program. All types of environment assignments are listed, including those made via environment files, command line flags, shell variables, and the environment database.

This preference can be used only from the command line.

Data Type	boolean
Default Value	FALSE

The enva preference produces the same list of preferences as the envp preference described in envp on page SSR-3-20 However, the preferences in this list appear in alphabetic order.

envp

Lists (in precedence order) the environment assignments accessible to the Sentinel program. All types of environment assignments are listed, including those made via environment files, command line flags, shell variables, and the environment database. The listing is ordered by the precedence of the preference assignments, with highest precedence assignments listed first. The standard preference enva produces the same list, but in alphabetic order. This preference can be used only from the command line.

Data Type	boolean
Default Value	FALSE

Because the list of environment assignments produced by this preference is comprehensive, it includes preferences with misspelled names, invalid values, and even shell variables that do not apply to Sentinel programs. This is particularly useful for diagnostic purposes because it can pinpoint assignment problems. For instance, when an incorrect preference name is used, it shows up in the envp listing, but not in the sconfirm listing. It is common for envp to be used in conjunction with the sconfirm and dconfirm standard preferences.

Figure 3-5 Sample envp Output

```
spsentinel-envp

------------ Precedence-Order Environment Settings List --------------
Name:                   Source:                     Value:
-------------------------------------------------------------------------
opnet_dir:              shell var                   /usr/opnet
SYS_DIR:                shell var                   /usr/opnet/sys
OBS_DIR:                shell var                   /usr/opnet/obs
REL_DIR:                shell var                   /usr/opnet
DISPLAY:                shell var                   unix,
                                                    0.0
OP_DIR:                 shell var                   /opnet/prod/opnet
EXINIT:                 shell var                   "set ai sw=4 ts=4"
HOST_TYPE:              shell var                   SUN4
PWD:                    shell var                   /tmp
LOGNAME:                shell var                   tsmith
PATH:                   shell var                   .,
                                                    /bin,
                                                    /usr/bin,
                                                    /usr/ucb,
                                                    /usr/etc,
                                                    /usr/opnet/sys/bin
USER:                   shell var                   tsmith
   •
mod_dirs:               env_db                      /usr/users/tsmith/op_models,
                                                    /usr/opnet/models/base
opnet_dir:              env_db                      /usr/opnet
-------------------------------------------------------------------------
%
```

geocentric_model

Specifies which geocentric coordinate system to use for the earth. The systems are:

- right-sphere—The earth is modeled as a sphere, on which the x-axis intersects the surface at 0°N 0°E, the y-axis at 0°N 90°E, and the z-axis at 90°N.
- left-sphere—The earth is modeled as a sphere, on which the x-axis intersects the surface at 0°N 90°E, the y-axis at 0°N 0°E, and the z-axis at 90°N (x and y are reversed from the right-sphere system).
- ellipsoid—The earth is modeled as an ellipsoid, using the right-sphere coordinate axes.

The value of this preference changes only at start-up.

Data Type	enum
Constraints	right_sphere, left_sphere, ellipsoid
Default Value	right_sphere

handle_exception

Specifies whether Sentinel should handle memory access and other exceptions. If FALSE, exception handling is left to the operating system.

This preference is independent of the preference handle_exception_extended.

The value of this preference changes only at start-up.

Data Type	boolean
Default Value	TRUE

handle_exception_extended

Specifies whether to activate extended exception handling.

If TRUE, OPNET adds additional exceptions handling around ets callbacks (client code) to detect memory access and other exceptions. Errors detected by extended exception handling are reported in the error log with the error type “<<Exception Detected>>”. The first time an error is detected during a session, a dialog box displays to inform the user of the potential issue.

If FALSE, exception handling is performed by either the handle_exception option or the operating system.

This preferences applies to the Windows operating systems only. On operating systems other than Windows, this preference is ignored. Additionally, this preference is independent of the preference handle_exception.

Data Type	boolean
Default Value	TRUE

help

Generates an alphabetical list of every preference the program can accept. This list includes the preference's name, type, default value, and purpose. This preference can be used only from the command line.

Data Type	boolean
Default Value	FALSE

The help preference is especially useful when you are trying to use a program that is not very familiar; the list of preferences introduces the range of options the program accepts. The following example shows part of the list produced by the help preference for the op_manfile program.

Figure 3-6 Sample help Preference List

```
% op_manfile -help

Built-in preferences of op_manfile:
(*) command-line only; (!) required

-----------Name----------- -----Type------ --Default--- --------Description--------
-all                        <boolean>       FALSE        apply to all instances (oth
-allow_spaces_in_file_names      <boolean>       FALSE        allow spaces in some m
-app_name                   <string>        ""           application name (used with
-arch_suffix                <boolean>       TRUE         enable architecture suffixes
-automation_lock_max_wait_time      <integer>       60           maximum time in mi
-backup_max_count           <integer>       10           maximum number of numbered
-beep_count_confirm        <integer>      1           audible beep count for confirm
-beep_count_error           <integer>       2            audible beep count for error
-buffered_tfile_types       <string list>   <empty>      list of tfiles that should
-checksum                   <boolean>       FALSE        compute checksum of file
-collect_model_deps_in_file  (*) <string>       ""           generate a list of model
-comments_add             <boolean>      FALSE       add comments to the model or file
-comments_del              <boolean>      FALSE       delete all comments from model
-comments_print            <boolean>       FALSE        print comments
-console_exit_pause        <boolean>       FALSE        wait for explicit input to
-dconfirm                (*) <boolean>       FALSE        print dynamic attribute con
-diag_enable               <boolean>       FALSE       enable the printing / logging
-dir                       <boolean>       FALSE        apply to all model files in
-edit_lock                 <boolean>       FALSE        set the edit lock
-edit_lock_mode          <boolean>     FALSE      enables use of edit locks on models
```

mem_clear

Specifies that the program should clear out an initial *n* megabytes of system memory before starting. This preference can be useful when debugging potential memory corruption problems.

This preference can be used only from the command line, and its value changes only at start-up.

Data Type	integer
Default Value	0

mem_optimize

Specifies whether Sentinel should use optimized memory allocation routines. By default, Sentinel uses several routines to optimize memory allocation. To suppress this feature, set the value of mem_optimize to FALSE. You may want to suppress memory optimization when running simulations along with certain debugging utilities and libraries, such as Purify.

This preference can be used only from the command line, and its value changes only at start-up.

Data Type	boolean
Default Value	TRUE

When mem_optimize is FALSE, the way OPNET handles memory allocation is limited:

- The concept of pooled memory does not exist; thus, all allocations are individual ones.
- Categorized memory is not used (although categories are defined).

Because of these memory allocation limitations, memory reports provided by OPNET (in the Simulation Sequence dialog box) will differ from those given when mem_optimize is TRUE, as follows:

- No pooled memory is listed.
- Categorized memory categories are listed, but all show a usage of zero.
- All memory usage is lumped together in one category, called General.
- Total usage is typically lower, because there is no pooled memory overhead.

mod_dirs

Specifies the full path names of the directories to be used for models and icon databases. This preference is not required by all Sentinel programs; however, if no value can be found in the environment, many programs will abort with an error.

Data Type	string
Default Value	“”

Many Sentinel programs perform operations on models and must know where models are stored in the host file system. In addition, GUI-based programs must know where to look for the icons used by the interface. Typically, the mod_dirs preference is assigned a value in the environment database (`env_db<rel>`), especially because the default contents of the environment database include this assignment.

Example mod_dirs assignments appear in the following diagram. Because this preference is a list of strings, individual elements of its value are separated by the appropriate delimiters.

Sentinel programs use the order in which directories are listed in mod_dirs to establish precedence among model files and icon databases with conflicting names. See Model Directory Organization on page SSR-2-16 for a description of this process and guidelines for listing model directories.

Figure 3-7 Sample mod_dirs Assignments

via an environment file:

```
# *** env_db ***

mod_dirs:   /usr/users/tsmith/op_models,
            /usr/opnet/11.5.A/models/std/base,
            /usr/opnet/11.5.A/models/std/ethernet,
            /usr/opnet/11.5.A/sys/icons
```

via a UNIX shell variable:

```
% setenv mod_dirs "/usr/users/tsmith/op_models:\
    /usr/opnet/11.5.A/models/std/base:\
    /usr/opnet/11.5.A/models/std/ethernet:\
    /usr/opnet/11.5.A/sys/icons"
% spsentinel
```

via a Windows shell variable:

```
C:\> set mod_dirs=\usr\users\tsmith\op_models:\
    \usr\opnet\11.5.A\models\std\base:\
    \usr\opnet\11.5.A\models\std\ethernet:\
    \usr\opnet\11.5.A\sys\icons
C:\> spsentinel
```

via a command line flag:

```
spsentinel -mod_dirs /usr/users/tsmith/op_models \
    /usr/opnet/11.5.A/models/std/base \
    /usr/opnet/11.5.A/models/std/ethernet \
    /usr/opnet/11.5.A/sys/icons
```

new_env_db

Causes the Sentinel program to create a new environment database file based on the current environment.

This preference can be used only from the command line, and must be the first argument of the Sentinel program.

Data Type	boolean
Default Value	FALSE

no_env_db

Causes the Sentinel program to ignore the user account's environment database. When used on the command line, this preference must be the first argument of the Sentinel program, or the first one after new_env_db.

Data Type	boolean
Default Value	FALSE

Sentinel programs obtain the values of preferences from several sources, including the user account's environment database. In certain cases, it is useful to ignore the environment database by setting no_env_db to TRUE. One such case occurs for the root user account, which does not have an environment database.

noprompt

Causes the Sentinel program to suppress prompts for the values of any unassigned dynamic preferences. Instead, all such unassigned preferences are automatically assigned the associated default values.

This preference can be used only from the command line.

Data Type	boolean
Default Value	FALSE

Dynamic preferences that are not assigned values in the program environment normally cause the program to print a prompt on the standard output device, requesting you to enter a value. This feature is especially useful in simulations, where values for promoted object attributes are sought from dynamic preferences, thus giving the simulation a "runtime" parameterization capability.

There are occasions, however, when a simulation will have several promoted object attributes to which you would rather not explicitly assign values. This situation usually arises during simulation debugging iterations, when the same simulation is run repeatedly to analyze a specific aspect of its execution. Under these conditions, it may be unnecessary to assign specific values to each promoted object attribute. The noprompt preference allows you to have default values assigned automatically to all dynamic preferences that would otherwise cause prompts.

opnet_dir

Specifies the full path name of the OPNET directory (<*opnet_dir*>) for Sentinel programs. This preference is not required by all Sentinel programs; however, if no value can be found in the environment, many programs will abort with an error.

Data Type	string
Default Value	""

Sentinel programs must have knowledge of where the Sentinel files are located in the host file system, because these programs may access system binary and ASCII data files while executing. Typically, opnet_dir is assigned a value in the environment database (`env_db11.5`), especially because the default contents of the environment database include this assignment.

Figure 3-8 Sample opnet_dir Assignments

via an environment file:

```
# *** env_db ***
opnet_dir:     /usr/opnet
```

via a UNIX shell variable:

```
% setenv opnet_dir /usr/opnet
% spsentinel
```

via a Windows shell variable:

```
C:\> set opnet_dir=\usr\opnet
C:\> spsentinel
```

via a command line flag:

```
spsentinel -opnet_dir /usr/opnet
```

opnet_user_home

Specifies the full path name of the user home directory. By default, Sentinel uses the path specified by a shell variable like $HOME or %HOMEDRIVE%/%HOMEPATH%; this preference lets you override the default.

This preference can be set only from the command line. If Sentinel is using the specified directory for the first time, it creates `/op_admin` and `/op_models` subdirectories and then runs the op_addlicense program.

Data Type	string
Default Value	“”

sconfirm

Prints a list of all preferences and their values. The list gives the name, the source of the current value, and the current value itself for each preference of the program. This preference can be used only from the command line.

Data Type	boolean
Default Value	FALSE

When using Sentinel, it is sometimes useful to confirm that its preferences are being set to appropriate values. This is especially a concern when several preference mechanisms are being used simultaneously, leaving some question as to the final value of a preference. The sconfirm preference prints a list of all preferences and their assigned values on the standard output shortly after the program starts executing. It does not otherwise affect execution of the program.

The sconfirm preference is often used in conjunction with the envp and enva standard preferences, which print the environment settings in precedence and alphabetic order, respectively.

The following example shows part of an sconfirm preference list. There are four sources given for the preference values:

- `cmd line`—The preference is set by a command line flag.
- `default`—The preference is not set by any environment mechanism (and thus has its default value).
- <env_file>—The preference is set by the named environment file, which either is specified by the -ef command line flag or is the environment database.
- `shell var`—The preference is set by a shell variable.

Figure 3-9 Sample sconfirm Preference List

```
% spsentinel-ef more_defs -sconfirm

------------ Static Environment Attribute Confirmations --------------
Name:                                Source:        Value:
----------------------------------------------------------------------
opnet_dir:                           shell var      /usr/opnet
mod_dirs:                            env. file   usr/users/tsmith/op_models,
                                                 /usr/opnet/models/base


 ef:                                 cmd line       more_defs
arch_suffix:                         default        TRUE
an_mono_sym:                         default        TRUE
backup_interval:                     default        15
backup_max_count:                    default        10
beep_enable:                         default        TRUE
bind_flags:                          default        <null>
code_area_height:                    env. file      410
code_area_width:                     env. file      500
comp_flags:                          default        <null>
core_dump:                           default        FALSE
dbl_click:                           default        250
diag_enable:                         env. file      TRUE
disp_host:                           default        unix
disp_server:                         default        0.0
  •
   •
   •
----------------------------------------------------------------------
%
```

In Figure 3-9, several preferences obtain their values from an environment file specified by the command line flag `-ef more_defs`. The contents of the file `more_defs.ef` appear below.

```
# more_defs.ef Environment File


diag_enable:               TRUE
code_area_height:          410
code_area_width:          500
```

Preference Sets

Most Sentinel programs share common preferences. These can be grouped into *preference sets* according to their use. For example, some preferences assist in tracking program behavior, while others deal with manipulating graphical options of the user interface. The preferences in these sets are described here and referred to in the discussions of programs that accept them.

The preference sets are listed in the following table.

Table 3-6 Preference Sets

Name	Coverage
Development	Software development utility options
Diagnostics	Error handling and memory usage
File	File handling and search paths
GUI	Graphical user interface options
Licensing	License System interface
Printing	Print output format and spooling options
Standard	Preference processing options (described in Standard Preferences on page SSR-3-16)
User Lock	User-initiated model write locking
Web Reporting	HTML report options
XML	XML import and export
End of Table 3-6	

All of these preference sets are described in this section except the standard set, which is described in Standard Preferences on page SSR-3-16.

Development

Development preferences are used in the process of developing software modules: e.g., object code files, archive-based object code libraries, or executable programs. These attributes manipulate the operating system commands that compile, archive, and bind (link). Only OPNET programs that create software modules accept these attributes, and some of these programs only accept a subset of the attributes (for instance, just the compile and archive-related attributes).

When compiling C++ code in OPNET, there are several development preferences that must be set to correctly compile and link the code. Table 3-7 specifies these attributes and their values for each operating system and compiler, depending on whether op_mksim or op_runsim is being used to create the simulation executable file.

Table 3-7 Compiling and Linking Program Preferences for C++

OS	Version	Compiler	op_mksim	op_runsim
Solaris	2.7 and later	CC (SunPro) 5.0 and later	comp_prog_cpp: comp_unix_cpp bind_static_prog: bind_unix	comp_prog_cpp: comp_unix_cpp bind_shobj_prog: bind_so_CC
		gcc[1] 2.x	comp_prog_cpp: comp_g++ bind_static_prog: bind_g++	comp_prog_cpp: comp_g++ bind_shobj_prog: bind_so_g++
		3.0 and later	comp_prog_cpp: comp_g++_3 bind_static_prog: bind_g++	comp_prog_cpp: comp_g++_3 bind_shobj_prog: bind_so_g++
Windows	NT 4.0, 2000, XP	Visual C++	comp_prog_cpp: comp_msvc bind_static_prog: bind_msvc	comp_prog_cpp: comp_msvc bind_shobj_prog: bind_so_msvc
End of Table 3-7				

1. When using gcc for C code, use these settings: comp_prog: comp_gcc, bind_static_prog: bind_gcc, bind_shobj_prog: bind_so_gcc

bind_shobj_flags

Specifies flags to be passed to the “binder” program when creating a shared object.

Data Type	string
Default Value	“”

bind_shobj_flags_devel

Specifies flags to be passed to the “binder” program when creating a shared object for a development simulation kernel.

Data Type	string
Default Value	“/DEBUG /DEBUGTYPE:BOTH”

bind_shobj_flags_optim

Specifies flags to be passed to the “binder” program when creating a shared object for an optimized simulation kernel.

Data Type	string
Default Value	“”

bind_shobj_libs

Specifies optional libraries to be passed to the “binder” program (i.e., the ECI translator that causes object code files to be linked into an executable program). The binder program is indirectly invoked through the ECI program specified by the bind_shobj_prog preference. The value of bind_shobj_libs should be a list of one or more library names separated by spaces. Library names can be simple file names from the current working directory or (preferably) full path names.

Data Type	string
Default Value	“”

bind_shobj_libs_devel

Specifies optional libraries to be passed to the “binder” program (i.e., the ECI translator that causes object code files to be linked into an executable program) for use with a development simulation kernel. The binder program is indirectly invoked through the ECI program specified by the bind_shobj_prog preference. The value of bind_shobj_libs_devel should be a list of one or more library names separated by spaces. Library names can be simple file names from the current working directory or (preferably) full path names.

Data Type	string
Default Value	“”

bind_shobj_libs_optim

Specifies optional libraries to be passed to the “binder” program (i.e., the ECI translator that causes object code files to be linked into an executable program) for use with an optimized simulation kernel. The binder program is indirectly invoked through the ECI program specified by the bind_shobj_prog preference. The value of bind_shobj_libs_optim should be a list of one or more library names separated by spaces. Library names can be simple file names from the current working directory or (preferably) full path names.

Data Type	string
Default Value	“”

bind_shobj_prog

Specifies the ECI translation program or script to invoke to create a shared object library.

Data Type	string
Default Value	“”

bind_static_flags

Specifies optional flags to be passed to the “binder” program when creating a static executable.

Data Type	string
Default Value	“”

bind_static_flags_devel

Specifies optional flags to be passed to the “binder” program when creating a static executable for a development simulation kernel.

Data Type	string
Default Value	“/Z7 /Od”

bind_static_flags_optim

Specifies optional flags to be passed to the “binder” program when creating a static executable for an optimized simulation kernel.

Data Type	string
Default Value	“”

bind_static_libs

Specifies optional libraries to be passed to the “binder” program (i.e., the ECI translator that causes object code files to be linked into an executable program). The binder program is indirectly invoked through the ECI program specified by the bind_static_prog preference. The value of bind_static_libs should be a list of one or more library names separated by spaces. Library names can be simple file names from the current working directory or (preferably) full path names.

Data Type	string
Default Value	“”

bind_static_libs_devel

Specifies optional libraries to be passed to the “binder” program (i.e., the ECI translator that causes object code files to be linked into an executable program) for a development simulation kernel. The binder program is indirectly invoked through the ECI program specified by the bind_static_prog preference. The value of bind_static_libs_devel should be a list of one or more library names separated by spaces. Library names can be simple file names from the current working directory or (preferably) full path names.

Data Type	string
Default Value	“”

bind_static_libs_optim

Specifies optional libraries to be passed to the “binder” program (i.e., the ECI translator that causes object code files to be linked into an executable program) for an optimized simulation kernel. The binder program is indirectly invoked through the ECI program specified by the bind_static_prog preference. The value of bind_static_libs_optim should be a list of one or more library names separated by spaces. Library names can be simple file names from the current working directory or (preferably) full path names.

Data Type	string
Default Value	“”

bind_static_prog

Specifies the ECI translation program or script to invoke to create static executables. Changing the default value of this preference may necessitate a change of the flags passed to the ECI program by the bind_static_flags preference.

Data Type	string
Default Value	“”

check_newer_process_model_files

Specifies which dates OPNET consults when deciding whether to recompile a file. if FALSE, only object file dates are checked. When TRUE, both object and process model file dates are checked.

Data Type	boolean
Default Value	TRUE

comp_flags_c++_specific

Specifies optional flags passed to a C/C++ compiler when compiling C++ code. These flags will be combined with the flags specified by other comp_flags_ preferences (comp_flags_common and others, as needed) and passed to the compiler program.

The value of this preference is compiler-specific and must have a format that is accepted by the compiler being used. For example, compiler flags are typically preceded by a dash.

Note—If a dash-preceded value for this preference is being specified on a command line, either the dash must be represented by a carat (-bind_static_flags ^g) or the value must be enclosed in double quotes, with a space character preceding the dash (for example, -bind_static_flags " -g").

Data Type	string
Default Value	""

comp_flags_c_specific

Specifies optional flags passed to a C/C++ compiler when compiling C code. These flags will be combined with the flags specified by other comp_flags_ preferences (comp_flags_common and others, as needed) and passed to the compiler program.

The value of this preference is compiler-specific and must have a format that is accepted by the compiler being used. For example, compiler flags are typically preceded by a dash.

Note—If a dash-preceded value for this preference is being specified on a command line, either the dash must be represented by a carat (-bind_static_flags ^g) or the value must be enclosed in double quotes, with a space character preceding the dash (for example, -bind_static_flags " -g").

Data Type	string
Default Value	""

comp_flags_common

Specifies optional compiler flags (such as include paths) to be used for all compilations. These flags will be combined with the flags specified by other comp_flags_ preferences (as needed) and passed to the compiler program.

The value of this preference is compiler-specific and must have a format that is accepted by the compiler being used. For example, compiler flags are typically preceded by a dash.

Note—If a dash-preceded value for this preference is being specified on a command line, either the dash must be represented by a carat (-bind_static_flags ^g) or the value must be enclosed in double quotes, with a space character preceding the dash (for example, -bind_static_flags " -g").

Data Type	string
Default Value	""

comp_flags_devel

Specifies optional compiler flags to be used when compiling code in debugging mode. These flags will be combined with the flags specified by other comp_flags_ preferences (comp_flags_common and others, as needed) and passed to the compiler program. The default value represents the debugging flags for Sun and Microsoft compilers.

The value of this preference is compiler-specific and must have a format that is accepted by the compiler being used. For example, compiler flags are typically preceded by a dash.

Note—If a dash-preceded value for this preference is being specified on a command line, either the dash must be represented by a carat (-bind_static_flags ^g) or the value must be enclosed in double quotes, with a space character preceding the dash (for example, -bind_static_flags " -g").

Data Type	string
Default Value	Windows: /Zi /Od UNIX: -g

comp_flags_mt

Specifies optional compiler flags to be used when compiling code in parallel mode. These flags will be combined with the flags specified by other comp_flags_ preferences (comp_flags_common and others, as needed) and passed to the compiler program.

The value of this preference is compiler-specific and must have a format that is accepted by the compiler being used. For example, compiler flags are typically preceded by a dash.

Note—If a dash-preceded value for this preference is being specified on a command line, either the dash must be represented by a carat (-bind_static_flags ^g) or the value must be enclosed in double quotes, with a space character preceding the dash (for example, -bind_static_flags " -g").

Data Type	string
Default Value	""

comp_flags_optim

Specifies optional compiler flags to be used when compiling code in optimized mode. These flags will be combined with the flags specified by other comp_flags_ preferences (comp_flags_common and others, as needed) and passed to the compiler program. The default value represents some optimization flags for Sun and Microsoft compilers.

The value of this preference is compiler-specific and must have a format that is accepted by the compiler being used. For example, compiler flags are typically preceded by a dash.

Note—If a dash-preceded value for this preference is being specified on a command line, either the dash must be represented by a carat (-bind_static_flags ^g) or the value must be enclosed in double quotes, with a space character preceding the dash (for example, -bind_static_flags " -g").

Data Type	string
Default Value	Windows: /Ox /Ob2 /DOPD_NO_DEBUG UNIX: -xtarget=ultra -xO3 -xbuiltin=all -xlibmil -DOPD_NO_DEBUG

comp_globals_export

When creating shared object libraries, specifies whether or not the library will require functions and variables declared in other libraries. If true, any variable declared with IMPORT_GLOBAL will look in other libraries for definition.

Required under Windows only.

Data Type	boolean
Default Value	FALSE

comp_per_process_errors

Specifies whether cc_err output files from each compilation should have a process ID appended to their names. This allows multiple compilations to be executed in parallel without the error files being overwritten.

Data Type	boolean
Default Value	TRUE

comp_prog

Specifies the name of the ECI translation program or script to invoke to compile C source code. Changing the value of this preference may necessitate a change of the flags passed to the C compiler program by the comp_flags_c_specific preference.

Data Type	string
Default Value	comp_unix

comp_prog_cpp

Specifies the name of the ECI translation program or script to invoke to compile C++ source code. Changing the value of this preference may necessitate a change of the flags passed to the C++ compiler program by the comp_flags_c++_specific preference.

Data Type	string
Default Value	comp_unix_cpp

comp_trace_info

Modifies compiled source code to include additional trace information.

Data Type	boolean
Default Value	TRUE

core_dump

Forces an OPNET program to retain its state in a core file in the current directory if the program encounters a program fault (segmentation violations, floating exceptions, or bus errors). These are typically caused by an improperly trapped condition (for instance, dividing by zero). When such errors occur, use the op_vuerr utility to see where the error occurred. If you contact OPNET by phone, you might be asked to run the UNIX program debugger dbx. The core dump created by this preference allows the error to be partially analyzed without having to recreate it while the program is running under dbx. Setting this preference does not affect the normal operation of OPNET programs.

This preference is usable only on UNIX platforms.

Data Type	boolean
Default Value	FALSE

debug_mk

Specifies that an OPNET program print to standard output the exact compiling and linking commands (including flags) it sends to the operating system. This can be useful when you have trouble compiling or linking process-model or other code.

Note—Printing the actual command will typically cause the top-level compiling or linking operation to fail. Therefore, you should have this preference set to TRUE only while performing diagnostics or debugging.

Data Type	boolean
Default Value	FALSE

ema_verbose

Specifies that EMA application code generated by an OPNET tool should include assignments for all object attributes, even those attributes with default values. Non-verbose EMA application code only includes assignment for attributes that have non-default values.

Data Type	boolean
Default Value	FALSE

kernel_type

Specifies which kernel is to be dynamically linked into simulations by op_runsim. Either four or eight kernel types are available, depending on platform:

- Windows and Solaris
  - *development*. This kernel allows ODB use, profiling, and detailed function call stack (FCS) information in errors and warnings.
  - *optimized*. This kernel runs much faster than development, but does not allow ODB use. Also, no profiling data is collected for kernel procedures, nor will any OPNET APIs be listed in the FCS information. Model code can still be profiled or can still include FCS information, if needed.
  - *development_parallel*. This kernel is similar to development, plus it allows multiple events to be executed in parallel.
  - *optimized_parallel*. This kernel is similar to optimized, plus it allows multiple events to be executed in parallel.
- Solaris only
  - *development_64bit*. This kernel is similar to development, plus it uses 64-bit addresses to give a larger space for executables.
  - *optimized_64bit*. This kernel is similar to optimized, plus it uses 64-bit addresses to give a larger space for executables.
  - *development_parallel_64bit*. This kernel is similar to development_parallel, plus it uses 64-bit addresses to give a larger space for executables.
  - *optimized_parallel_64bit*. This kernel is similar to optimized_parallel, plus it uses 64-bit addresses to give a larger space for executables.

Depending on the value of kernel_type, the op_runsim program will invoke one of the following programs to link and run simulations:

- op_runsim_dev
- op_runsim_opt
- op_runsim_mtdev
- op_runsim_mtopt
- op_runsim_dev64
- op_runsim_opt64
- op_runsim_mtdev64
- op_runsim_mtopt64

This preference is also used by op_mkso to determine which type of shared object file to create, by op_mko to build appropriate object files, and by op_mksim to build static simulations.

Data Type	string
Default Value	development

mem_track

Specifies that the memory allocation and deallocation tracking features of the program should be enabled for use within the host workstation debugger. If mem_track is enabled, internal program functions can be invoked to print lists of the allocation sources and deallocation sinks within the program. This is accomplished by invoking the function *Vos_Memstats_Print_Sources* ***(***obj_name***,*** mode***)*** where the string argument obj_name is the name of a particular memory object and mode is either 0 (for a list of allocation sources) or 1 (for a list of deallocation sinks). To obtain a list of significant memory object names, invoke the function *Vos_Memstats_Print_Table (1, 2)*. If mem_track is disabled, these functions are not functional.

Data Type	boolean
Default Value	FALSE

mem_track_object

Specifies the name of a particular object to track memory allocation and deallocation. Specifying an object allows for efficient tracking if only one object is of interest. Used when mem_track is enabled.

Data Type	string
Default Value	“”

ov_diff_tolerance

Specifies a tolerance, expressed as a percentage value, to use when testing the equality of graphs. Vectors whose mean values differ by less than this tolerance will be considered equal.

Data Type	integer
Default Value	5

ov_mem_size

Specifies the maximum size of the output vector data buffer in megabytes. When a buffer reaches the specified limit, its entries are written to disk, and it resumes the collection of new values. Setting this preference to larger values than the default trades off less writes to disk versus more memory usage. If fragments are allowed to grow too large, the simulation may need to use swap space which will reduce execution speed.

Type	integer
Default Value	4

prof_output

Specifies a file name or other destination to which function profiling information should be sent.

Data Type	string
Default Value	“console”

prof_output_separator

Specifies text to be placed between function profile fields.

Data Type	string
Default Value	“”

prof_track

Specifies that function profiling information be reported where possible. When a simulation is run, OPNET reports the number of times each function was called, and how many seconds were spent within that function.

Data Type	boolean
Default Value	FALSE

prof_track_func

Specifies a single function for which to report tracking information (if prof_track is enabled)

Data Type	string
Default Value	“”

prof_track_recurse

Collects profiling information on recursive function calls (if prof_track is enabled).

Data Type	boolean
Default Value	TRUE

progress_bar_positioning

Specifies the positioning of progress bars. If “recenter”, progress bars are kept centered on the current window. If “fixed”, they remain in a fixed global position.

Data Type	enum
Constraints	recenter, fixed
Default Value	“recenter”

show_compilation_output

Specifies whether OPNET should display the output of each compilation, even if successful. When TRUE, compilation output is always displayed.

Data Type	boolean
Default Value	FALSE

show_compilation_success_dialog

Specifies whether OPNET should display a dialog box listing the expected targets and the compilation status of each.

Data Type	boolean
Default Value	TRUE

verbose_mk

Specifies that all commands issued to compile and bind code be displayed.

Data Type	boolean
Default Value	FALSE

Diagnostics

Diagnostics preferences control the handling of program errors and provide additional diagnostic services. All Sentinel programs accept diagnostic preferences, thus letting you make error handling consistent throughout Sentinel by placing value assignments for these preferences in your environment database.

beep_count_confirm

Specifies the number of beeps generated to confirm completion of selected operations (typically long operations that may result in failure). To disable confirmation beeps altogether, set this preference to 0.

Data Type	integer
Default Value	1

beep_count_error

Specifies the number of beeps generated to indicate an error condition. To disable error beeps altogether, set this preference to 0.

Data Type	integer
Default Value	2

diag_enable

When TRUE, specifies that diagnostic errors are to be detected, printed (either on the standard output device or within the program), and logged (in the `<HOME>/op_admin/err_log` file). Diagnostic errors indicate program conditions that are incorrect, but recoverable, and that are not normally checked

during program execution. (Typically, diagnostic error conditions are rare and automatic checking would reduce performance excessively.) It is useful to enable this preference during development and debugging sessions to ensure notification of all errors.

Data Type	boolean
Default Value	FALSE

rec_err_suppress

When TRUE, specifies that recoverable errors and warnings should be suppressed from printing (either on the standard output device or within the program). Even when printing is suppressed, recoverable errors and warnings continue to be detected and logged in the `<HOME>/op_admin/ err_log` file and can be viewed using the op_vuerr utility. Enabling this preference is useful when programs generate many expected errors.

Data Type	boolean
Default Value	FALSE

warning_suppress

Specifies that warnings should be suppressed from printing (either on the standard output device or within the GUI). If warning_suppress is enabled, warnings are still detected and logged in the `<HOME>/op_admin/err_log` file, and can be viewed using the op_vuerr utility. Enabling this preference is useful in situations where programs generate many expected warnings.

Data Type	boolean
Default Value	FALSE

File

File preferences indicate options that affect file handling, backups, and search paths. Several standard preferences appear among the file preferences when you edit program preferences—arch_suffix, ef, opnet_dir, and mod_dirs. These are described in Standard Preferences on page SSR-3-16.

backup_interval

Specifies the length of the interval between automatic backups of Sentinel models. Setting this preference's value very high will effectively prevent automatic backups from occurring during the session.

Data Type	integer
Default Value	15
Units	minutes

backup_max_count

Specifies the maximum quantity of numbered version backups kept for a given file. Each time you save a model with the same name, the overwritten file is stored in your op_admin/bk directory with a name of the form: `<model_name>.v<vers_num>.<mdl_type>.m`. The most recent file versions are kept; when the maximum count of backups would be exceeded by a new version, the earliest version is removed from the bk directory.

Data Type	integer
Default Value	10
Units	versions

buffered_tfile_types

Specifies a list of tfile types and whether or not each type should be buffered. The preference value should be a comma-separated list of file extensions. Each file type listed will be buffered unless the extension is preceded by an exclamation point (!). For example,

```
".ot, .otf, !.sddb"
```

means that output table and output table fragment files should be buffered and self-description database file should not be buffered.

Data Type	string list
Default Value	<empty>

file_read_abort_suppress

Specifies whether OPNET should suppress model file reading errors. When TRUE, such errors are suppressed.

Data Type	boolean
Default Value	FALSE

force_posix_locale

Specifies whether OPNET should run in the POSIX locale. When TRUE, POSIX locale is forced.

The value of this preference changes only at startup.

Data Type	boolean
Default Value	FALSE

verbose_session_log

Specifies whether to use verbose session logging. When TRUE, verbose session logging is enabled.

Data Type	boolean
Default Value	FALSE

GUI

Graphic user interface (GUI) preferences specify options that affect the graphical user interface of programs, including display device parameters, window appearance, and interaction styles. They also provide control of shell scripts and batch files.

Several dynamic preferences appear among the GUI preferences when you edit program preferences. These are described in Static vs. Dynamic Preferences on page SSR-3-12.

allow_rotated_world_coordinates

Allows subnets with degree units and 360 degree spans to wrap in the x direction. If TRUE, subnets can span the International Date Line (180 degrees longitude). If FALSE, subnets cannot span the International Date Line. Instead, subnets must be placed entirely on either one side of the International Date Line or the other side.

Data Type	boolean
Default Value	FALSE

annot_fonts.large

Specifies the font used for large text annotations.

Data Type	string
Default Value	“family:Lucida Console-size:9”

annot_fonts.medium

Specifies the font used for medium text annotations.

Data Type	string
Default Value	“family:Lucida Console-size:8”

annot_fonts.small

Specifies the font used for small text annotations.

Data Type	string
Default Value	“family:Lucida Console-size:7”

async_cmd_mode

Specifies that the program should accept asynchronous commands. Currently, spsentinel is the only program that supports this preference.

The value of this preference changes only at start-up.

Data Type	boolean
Default Value	FALSE

async_cmd_port

Specifies a port for asynchronous commands. Currently, spsentinel is the only program that supports this preference.

The value of this preference changes only at start-up.

Data Type	integer
Default Value	3567

async_cmd_trace

When TRUE, specifies that tracing of asynchronous commands should be enabled. Currently, spsentinel is the only GUI program that supports this preference.

Data Type	boolean
Default Value	FALSE

browser_prog

Specifies the full path name of an HTML browser program for viewing web reports.

Data Type	string
Default Value	""

check_subnet_bounds

Specifies whether to enforce subnet bounds checking on the positions of contained sites. When TRUE, subnet bounds are checked.

Data Type	boolean
Default Value	TRUE

code_editor_prog

Specifies the full path name of the text editor program to use for editing C/C++ source code. The special value “editor_prog” means that the text editor specified by the editor_prog preference should be used.

Data Type	string
Default Value	“editor_prog”

console

When TRUE, specifies that the application console should be used for stdout and stderr messages.

Data Type	boolean
Default Value	FALSE

creation_mode

Specifies the mode for creating objects with the mouse. "repeat" allows the user to continue to create instances of the object until the operation is ended with a right-click. "single" automatically ends the operation after only one object is created.

Data Type	enum
Default Value	repeat

disable_message_confirmations

When TRUE, specifies that OPNET should not display interactive message confirmations.

Data Type	boolean
Default Value	FALSE

disable_message_tips

When TRUE, specifies that OPNET should not display pop-up message tips.

Data Type	boolean
Default Value	FALSE

disable_object_operations

When TRUE, specifies that OPNET should not show operation buttons in the Advanced Object Attributes editor.

Data Type	boolean
Default Value	FALSE

disable_object_palette_configuration

When TRUE, specifies that OPNET should not show the Configure button in object palettes.

Data Type	boolean
Default Value	FALSE

disable_object_palette_model_properties_display

When TRUE, specifies that OPNET should not display model details when an icon is right-clicked in the object palette.

Data Type	boolean
Default Value	FALSE

display

Specifies an identifier for a display host and server for remote graphics. The GUI-based program will send its X Window System graphics requests to the server indicated by this preference. The X server must be configured to permit graphics access from the host running the GUI-based program (this can be checked and modified via the X Window System xhost program). The host computer running the GUI-based program must be able to reach the display host via the IP network protocol.

The value of this preference must be of the form: <hostname>:<display_number>.<screen_number>. The hostname can be an IP address or a configured machine name. The display number and screen number are zero-based integers and are usually zero.

Data Type	string
Default Value	“:0.0”

doubleclickinterval

Specifies the maximum time, in seconds, between two mouse clicks for the clicks to be interpreted as a “double click” rather than as two “single clicks.” Users with fast input styles can set this preference to a shorter interval to avoid false double-clicks. Users with slow input styles can set this preference to a longer interval to ensure that slow double-clicks are detected.

Data Type	integer
Default Value	0.25

drag_obj_threshold

Specifies the maximum number of objects to show while dragging. When the number of objects being dragged is less than this value, a separate drag outline is drawn for each object. When the number is equal to or greater than this value, only two outlines are drawn—a small one representing the object the cursor is on and a large rectangle encompassing all of the dragged objects.

Data Type	integer
Default Value	32

edit_pad.default_height

Specifies the initial height of OPNET text editor windows, in pixels.

Data Type	integer
Default Value	480

edit_pad.default_width

Specifies the initial width of OPNET text editor windows, in pixels.

Data Type	integer
Default Value	640

edit_pad.drag_and_drop

Specifies whether drag-and-drop editing is used by the built-in text editor. When TRUE, drag-and-drop editing is permitted.

Data Type	boolean
Default Value	FALSE

edit_pad.font

Specifies the font used by the built-in text editor.

Data Type	string
Default Value	Windows: “family:Lucida Console-size:8” UNIX: “family:courier-face:medium-size:12”

edit_pad.selection_background_color

Specifies the background color for selected text in the built-in text editor.

Data Type	string
Default Value	RGB000

edit_pad.selection_color

Specifies the text color for selected text in the built-in text editor.

Data Type	string
Default Value	RGB333

edit_pad.show_line_number

Specifies whether the built-in text editor shows line numbers for the displayed text. When TRUE, line numbers are shown.

Data Type	boolean
Default Value	TRUE

edit_pad.text_background_color

Specifies the background color for text in the built-in text editor.

Data Type	string
Default Value	RGB333

edit_pad.text_color

Specifies the color of text in the built-in text editor.

Data Type	string
Default Value	RGB000

edit_pad.word_wrap

Specifies whether the built-in text editor should wrap lines of text when reaching the edge of the display area. When FALSE, lines are not wrapped and thus continue out of sight.

Data Type	boolean
Default Value	FALSE

editor_config

Specifies the name of the editor configuration file used by the program. This file defines the operations and menu layouts for the program.

The value of this preference changes only at start-up.

Data Type	string
Default Value	""

editor_keep

Specifies how to handle temporary files used to transfer text between the program and an external text editor such as vi, emacs, or notepad. When FALSE, the files are automatically removed when the text editor quits. When TRUE, the files are kept in the user's `<op_admin>/tmp` directory.

Data Type	boolean
Default Value	FALSE

editor_prog

Specifies a text editor to be used when editing text files other than source code. (Use the code_editor_prog preference to specify a source code editor.) The default text editor is the Sentinel text edit pad. To use an external editor such as vi, emacs, or Notepad, set this preference to the path name of the editor to be used.

If an external editor is specified, the program invokes it with one argument, which is the name of a file containing the text to be edited. The modified text must be in this same file when the text editor quits.

Sentinel includes two shell scripts for UNIX (vi_exec, and emacs_exec) and one for Windows (notepad_exec.bat) that allow you to use external text editors within Sentinel programs.

Data Type	string
Default Value	"builtin"

expanded_subnets_iconic

When FALSE, expanding a subnet removes the subnet icon and creates a bounding box based on the subnet's extent. When TRUE, the subnet icon remains in place and no bounding box appears. (Note that an object in the expanded subnet may overlap the icon, so it may not be visible.)

Data Type	boolean
Default Value	FALSE

expanded_subnets_thickness

Specifies the thickness of the borders of expanded subnets, in pixels.

Data Type	integer
Default Value	2

flow_spreadsheet_import_filename

Stores the default filename used when importing spreadsheet traffic.

Data Type	string
Default Value	“”

flow_spreadsheet_import_overwrite

Stores the default choice for overwriting traffic when importing spreadsheet traffic. When FALSE, only the traffic specified in the import file is overwritten. When TRUE, all traffic is replaced by the imported traffic.

Data Type	boolean
Default Value	FALSE

font

Specifies a default font used by various UI components.

Data Type	string
Default Value	“”

html_export_max_image_width

Specifies the maximum width, in pixels, for exported HTML pages.

The value of this preference changes only at startup.

Data Type	integer
Default Value	1024

http_port

Specifies the port number for ASN and web hot links. Set to -1 to disable.

The value of this preference changes only at startup.

Data Type	integer
Default Value	81

icon_autosizing.disable

Specifies whether automatic resizing of site icons in the workspace is enabled or disabled. When FALSE, autosizing is enabled.

Data Type	boolean
Default Value	FALSE

icon_autosizing.large_element_count

Specifies a threshold for disabling icon autosizing. When the number of nodes in a subnet exceeds this threshold, icon autosizing is disabled to avoid excessive screen-redraw delays.

Data Type	integer
Default Value	500

icon_autosizing.max_neighbors

Specifies the maximum number of overlapping neighbors OPNET will consider before minimizing an icon. This number is interpreted loosely, but increasing it will make automatic minimization less likely, and vice versa.

Data Type	integer
Default Value	30

icon_dbs

Specifies which icon databases should be used by Sentinel GUI-based programs. Although you could specify icons for all such programs at once, the programs will start up faster if you specify icons separately for each program. For example, to specify icon databases for op_vuanim in the environment database, you would add the line:

```
op_vuanim.icon_dbs: sys, op_vuanim
```

Any number of icon databases can be listed, including user-created or user-modified ones. Programs search the databases for an icon in the sequence in which the databases are listed and use the first instance of the icon found.

The value of this preference changes only at start-up.

Data Type	string
Default Value	""

image_capture.border_width

Specifies how wide the black border placed around image captures should be.

Data Type	integer
Default Value	0

image_capture.include_window_title

Specifies whether to include the window title when a window image is captured. When TRUE, the title is included.

Data Type	boolean
Default Value	TRUE

image_capture.screen_hot_key

Specifies a special key combination to start an image capture of a selected area of a window.

Data Type	string
Default Value	ctrl-shift-t

image_capture.show_in_browser

Specifies whether an image of a window or screen rectangle should be displayed in a browser after capture. When TRUE, the captured image is displayed.

Data Type	boolean
Default Value	FALSE

image_capture.window_hot_key

Specifies a special key combination to start an image capture of the active window.

Data Type	string
Default Value	ctrl-t

load_image

Specifies an image to display while the application starts.

This file must be converted to the OPNET image format, placed in the top mod_dirs (op_models) directory, and have a `.image` file extension.

The value of this preference changes only at startup.

Data Type	string
Default Value	“”

mark_nondefault_attrs

Specifies whether OPNET should use color in the Edit Attributes dialog box to highlight attributes that do not inherit their default value. When TRUE, attributes whose values are Intended or Changed have a colored background.

Data Type	boolean
Default Value	FALSE

mark_nondefault_attrs.changed_color

Specifies the color used to highlight values that have been changed in the current instance of an Edit Attributes dialog box.

Data Type	string
Default Value	#FFDFBF

mark_nondefault_attrs.intended_color

Specifies the color used to highlight values that do not inherit their attribute's default value, but which have not been changed in the current instance of an Edit Attributes dialog box.

Data Type	string
Default Value	#BFBFFF

model_name

Specifies the name of a model to initially open when an application is invoked. The value of this preference changes only at start-up.

Data Type	string
Default Value	""

move_delta

Specifies the number of pixels the mouse must move to initiate a drag and drop operation. This preference allows you to set the sensitivity of objects to being moved (dragged). Too small a value for this preference may allow an object to move slightly when your only intention was to click or double-click the object.

Data Type	integer
Default Value	10
Units	pixels

netbiz_show_promote_button

Specifies whether a Promote button is displayed in the attribute dialog boxes of a Netbiz application. When TRUE, the Promote button is displayed.

Data Type	boolean
Default Value	FALSE

network_add_world_map_to_new_models

Specifies whether a world map should be included in the top subnet of newly created network models. When TRUE, the map is automatically included.

Data Type	boolean
Default Value	TRUE

network_layout.adjust_unconnected_groups

Specifies whether the spacing of unconnected groups of nodes in a subnet should be adjusted. This preference applies to automatic layout operations using the balanced algorithm (such as during a topology import). Setting this preference to FALSE turns off adjustment of unconnected groups and might speed up the layout operation.

Data Type	boolean
Default Value	TRUE

network_layout.max_overlap_links_to_bend

Specifies the maximum number of overlapping links between nodes that will be represented with bends by an automatic balanced layout operation (such as during a topology import). The default value of –1 means no limit.

Data Type	integer
Default Value	–1

network_layout.use_simple_threshold

Specifies the number of sites in a subnet that will trigger the use of a simple, but efficient, layout algorithm instead of the more balanced algorithm. Topology imports use this preference to determine which layout algorithm to use.

Data Type	integer
Default Value	100

network_links_infer_threshold

Specifies whether OPNET should scale link thickness using a threshold inferred from the thresholds of attached nodes. When TRUE, link thickness will be narrowed as the icons of the attached nodes are scaled down during zoom operations.

Data Type	boolean
Default Value	FALSE

network_palette.style

Specifies whether the default display of the Object Palette is the tree or icon view.

Data Type	enum
Constraints	treeview, iconview
Default Value	treeview

network_path_add_subnet_single_left_click

Specifies whether subnets can be added to a path object with a single left click, rather than from a right-click menu. When TRUE, the left-click method is enabled.

Data Type	boolean
Default Value	FALSE

network_positions_in_deg_min_sec

Specifies how the Project Editor displays degree-position attributes when a subnetwork's grid properties are expressed in degrees. If TRUE, degrees are displayed in degrees-minutes-seconds format (e.g., 39 52' 46.30"). If FALSE, degrees are displayed in decimal format (39.8795). You can set an object's x position and y position attributes using either format, regardless of this preference's setting.

Data Type	boolean
Default Value	FALSE

network_show_links_to_parent_subnet_objects

Specifies whether OPNET should show links from an object in one subnet to objects in parent and sibling subnets. When TRUE, such links are shown.

Data Type	boolean
Default Value	TRUE

network_use_trx_prompt

Specifies whether OPNET should open the transceiver/port chooser dialog box when a user is manually hooking up links. When TRUE, the transceiver/port chooser is opened. When FALSE, the transceivers and port are chosen automatically.

Data Type	boolean
Default Value	TRUE

network_visualization.minimized_icon_pixel_threshold

Specifies the size threshold (in pixels) at which icons should be minimized. If an icon is scaled to this size or smaller, the icon will be drawn according to the site's "minimized icon" preference.

Data Type	integer
Default Value	12

network_visualization.minimized_icon_size

Specifies the size of a minimized icon, in pixels.

Data Type	integer
Default Value	6

network_visualization.show_demand_labels

Specifies whether demands are labeled when drawn in the workspace.

Data Type	enum
Default Value	never
Constraints	always, never

network_visualization.show_labels_icon_pixel_threshold

Specifies the size threshold (in pixels) for labeling nodes. If a node's icon is larger than this size and the network_visualization.show_node_labels preference is set to "for_icons_larger_than_specified", the node's label is displayed. To change the minimum icon size, go to the View menu and choose Edit Visualization Preferences... The Edit Visualization Preferences dialog box appears. Select the Icons larger than radio button and change the number in the text box to the minimum icon size you want.

Data Type	integer
Default Value	12

network_visualization.show_link_arrowheads

Specifies which links are drawn with arrowheads.

Data Type	enum
Default Value	for_simplex_links_only
Constraints	always, for_simplex_links_only, never

network_visualization.show_link_labels

Specifies whether links are labeled when drawn in the workspace.

Data Type	enum
Default Value	never
Constraints	always, for_off_page_links_only, never

network_visualization.show_node_labels

Specifies whether nodes are labeled when drawn in the workspace. The size for the “for_icons_larger_than_specified” option is specified by the network_visualization.show_labels_icon_pixel_threshold preference.

Data Type	enum
Default Value	for_non_minimized_icons
Constraints	always, for_non_minimized_icons, for_full_icons, for_icons_larger_than_specified, never

network_visualization.show_path_labels

Specifies whether paths are labeled when drawn in the workspace.

Data Type	enum
Default Value	never
Constraints	always, never

obj_dbox_headers

Specifies text strings to be displayed in the Edit Attributes dialog box of objects. Each string contains four pieces of information, separated by commas:

- <label_text>—A label indicating the value displayed, such as “Type ”or “Model: ”
- <label_tooltip>—The text that appears in the tooltip when a user rests the cursor over the label.
- <value_tooltip>—The text that appears in the tooltip when a user rests the cursor over the label.
- <value_text>—The data that appears in the value field. You can specify data from the model’s attribute list and self-description for this field.

For details about the text strings used in this preference, see Customizing Header Information on page SSU-9-25 in the *User Guide*.

Data Type	string list
Default Value	<empty>

orbit_draw_style

Specifies whether satellite orbits are drawn as dashed or solid lines.

Data Type	enum
Default Value	dashed
Constraints	dashed, solid

panels_close_behavior

Specifies the behavior of the close box in analysis panels, as follows:

- query—Clicking the close box opens a dialog giving the option to hide or remove the panel, or cancel the operation.
- hide—Clicking the close box hides the panel.
- remove—Clicking the close box removes the panel.

Data Type	enum
Default Value	query
Constraints	query, hide, remove

panels_tile_columns_size

Specifies the number of columns to use when tiling analysis panels.

Data Type	integer
Default Value	1

panels_tile_rows_size

Specifies the number of rows to create when tiling analysis panels.

Data Type	integer
Default Value	1

panels_tile_sizing_automatic

Specifies how analysis panels are sized when being tiled. When TRUE, panel sizes are calculated based on the number of panels and the window size. When FALSE, panels are tiled using the sizes specified by panels_tile_x_size and panels_tile_y_size.

Data Type	boolean
Default Value	TRUE

panels_tile_spacing

Specifies the desired spacing between tiled analysis panels, in pixels.

Data Type	integer
Default Value	1

panels_tile_x_size

Specifies the desired horizontal size of tiled analysis panels, in pixels.

Data Type	integer
Default Value	35

panels_tile_y_size

Specifies the desired vertical size of tiled analysis panels, in pixels.

Data Type	integer
Default Value	35

program_banner

Specifies the text that appears as the title of the program window. The title is usually taken from the program's configuration file, which will be overridden if this preference is given a value.

The value of this preference changes only at start-up.

Data Type	string
Default Value	""

program_resources

Specifies program resources (`.wrc` files) to be used in addition to those specified in the `<product>.ets` file.

The value of this preference changes only at start-up.

Data Type	string list
Default Value	<empty>

project_backup_on_dont_save

Specifies whether OPNET should back up project files when exiting the Project Editor. When TRUE, projects are backed up regardless of whether they were saved. When FALSE, projects are backed up only if saved.

Data Type	boolean
Default Value	FALSE

project_editable_attributes

Specifies a list of attributes that can be modified even if a project is read-only.

Data Type	string list
Default Value	<empty>

project_readonly

Specifies whether projects should be opened as read-only. When TRUE, projects cannot be modified (except for attributes listed by project_editable_attributes).

Data Type	boolean
Default Value	FALSE

recent_files_limit

Specifies the maximum number of recent files to list for each type of editor in the File > Recent Files list.

Data Type	integer
Default Value	5

report_top_n

Specifies the number of objects that will be included in the top results reports (including SLA reports).

Data Type	integer
Default Value	5

scenario_name

Specifies the name of a scenario to open initially in the Project Editor. (This is only useful if the model_name preference has been supplied and the Project Editor is used to open the model.)

The value of this preference changes only at start-up.

Data Type	string
Default Value	“”

select_live_network

When TRUE, objects are selected dynamically as you drag the mouse in the Project Editor workspace. When FALSE, objects are not selected until you release the mouse button.

Data Type	boolean
Default Value	TRUE

self_desc_validate

Specifies whether self-descriptions should be validated. When TRUE, Sentinel validates all self-descriptions before importing a network. When FALSE, Sentinel will trust the information currently stored in the self-description database, even though it might be out-of-date with respect to node model changes, additions, and removals.

Data Type	boolean
Default Value	TRUE

show_ca_count

Specifies whether the Count column should be displayed in compound object attribute dialog boxes.

Data Type	boolean
Default Value	FALSE

show_network_browser_threshold

Specifies the minimum number of nodes required in the project for the Network Browser to automatically display after an import.

Data Type	integer
Default Value	250

show_units

Specifies whether the Units column should be displayed in object attribute dialog boxes.

Data Type	boolean
Default Value	FALSE

sim_duration_default

Specifies the default duration for simulations, in hours.

Data Type	double
Default Value	1

splash_image

Specifies the image be display in the main application window.

This file must be converted to the OPNET image format, placed in the top mod_dirs (op_models) directory, and have a `.image` file extension.

The value of this preference changes only at start-up.

Data Type	string
Default Value	“”

subnet_hierarchy.interactive.reparenting_disable

Disables the ability to move nodes from one subnet to another subnet using either drag and drop or right-click operations.

Data Type	boolean
Default Value	FALSE

suppress_use_startup_wizard_checkbox

Specifies whether the checkbox “Use Startup Wizard when creating new scenarios” appears on the dialog box when specifying the name of a new scenario.

Data Type	boolean
Default Value	FALSE

This preference is used with the preference use_startup_wizard on page SSR-3-80.

system_window_behavior

Specifies how the system (main application) window should behave. The following behaviors are available:

- show_if_last—the system window will unhide and come to the front when the last editor window is closed, thus allowing the user to open a new editor window.
- quit_if_hidden—the application will quit without showing the system window when the last editor window is closed. The user must restart the application to open a new editor.
- hide_on_startup—same as quit_if_hidden, plus the application will start up with the system window hidden if an editor has been opened via a preference setting.
- hide_when_inactive—same as hide_on_startup, plus the system window will hide automatically whenever an editor is opened. The system window must be explicitly opened via the Windows > System Window operation.

Data Type	enum
Default Value	show_if_last

title_autoplacing.directions

Specifies allowable directions (relative to an icon) for placing a site label.

Data Type	enum
Default Value	bottom/top/right/left
Constraints	bottom/top/right/left, bottom/top, bottom/right/left, bottom

title_autoplacing.disable

Specifies whether automatic placement of site labels in the workspace is enabled or disabled. When FALSE, autoplacement is enabled.

Data Type	boolean
Default Value	FALSE

title_autoplacing.try_small_font

Specifies whether automatic placement of site labels can try to use a smaller font to make a label fit. When FALSE, a smaller font is not tried.

Data Type	boolean
Default Value	FALSE

tool_name

Specifies the name of an OPNET tool (such as the Project Editor) to initially open when the program starts.

The value of this preference changes only at start-up.

Data Type	string
Default Value	“”

tool_open_save_behavior

Specifies whether a file chooser should be used when editors are opened and saved. If used, the starting directory will be specified by the tool_open_start_dir preference. The following behaviors are available:

- browse_button_some—the Open and Save dialog boxes include a Browse button, for only those editors so specified in the spsentinel.ets file.
- browse_button_all—the Open and Save dialog boxes include a Browse button, for all editors.
- never_browse—never include a Browse button in the Open and Save dialog boxes, regardless of any spsentinel.ets file specification.

- always_browse—automatically open a file browser without providing the standard Open or Save dialog boxes.

Data Type	enum
Default Value	browse_button_some

tool_open_start_dir

Specifies the starting directory for the file browser opened when editors are opened and saved (if specified by the tool_open_save_behavior preference).

The value of this preference changes only at start-up.

Data Type	string
Default Value	“”

tool_override_dir_behavior

Specifies whether typed files in the same directory as a browsed-to model file also should be included in the FIles window of the Open dialog box. The following behaviors are available:

- same_name—typed files with the same base name are included.
- specific_file—only the browsed-to file is included.
- all_files—all typed files in the directory are included (these files override any identically named files in other mod_dirs directories).

The value of this preference changes only at start-up.

Data Type	enum
Default Value	same_name

tooltip.annot_attributes

Specifies the list of attributes to show in tool tips for annotations.

Data Type	string list
Default Value	<empty>

tooltip.demand_attributes

Specifies the list of attributes to show in tool tips for demands.

Data Type	string list
Default Value	<empty>

tooltip.link_attributes

Specifies the list of attributes to show in tool tips for links.

When a new environment database file is created by the new_env_db attribute, this preference is given the value "tool tip, name".

Data Type	string list
Default Value	<empty>

tooltip.object_update

Specifies whether to determine all object attributes before building the tool tip for an object. If this preference is set to FALSE, then occasionally incorrect or outdated information may appear in an object's tool tip. Setting this value to TRUE may result in slower performance.

Data Type	boolean
Default Value	FALSE

tooltip.path_attributes

Specifies the list of attributes to show in tool tips for paths.

Data Type	string list
Default Value	<empty>

tooltip.site_attributes

Specifies the list of attributes to show in tool tips for sites (fixed nodes).

When a new environment database file is created by the new_env_db preference, this preference is given the value "tooltip".

Data Type	string list
Default Value	<empty>

tooltip_delay

Specifies the delay after the cursor stops moving before the tool tip display appears.

The value of this preference changes only at start-up.

Data Type	integer
Default Value	1000
Units	milliseconds

tutorial_top

Specifies whether or not the tutorial window is above all other windows on the desktop, regardless of whether or not it is the active window.

This preference has no effect in the UNIX environment.

Data Type	boolean
Default Value	TRUE

ui.hide_all_windows

Specifies whether Sentinel should display any windows. When TRUE, all windows are hidden while the program runs.

Note—Because no windows are visible, Sentinel cannot be shut down normally. You must use an automation script, the Windows Task Manager, or the UNIX `kill` command to stop the program.

This preference can be used only from the command line, and its value changes only at start-up.

Data Type	boolean
Default Value	FALSE

ui.remember_window_sizes_and_positions

Specifies whether application windows should open with the same size and position they had last. When TRUE, size and position are remembered. When FALSE, windows open with their default size and position.

Data Type	boolean
Default Value	TRUE

ui_colors.tool_back

Specifies the background color of tool windows.

The value of this preference changes only at start-up.

Data Type	string
Default Value	RGB222 (light gray)

ui_colors.tool_fore

Specifies the color for foreground objects in tool windows, such as selection markers and the text used for node names and grid numbering.

The value of this preference changes only at start-up.

Data Type	string
Default Value	RGB000 (black)

ui_colors.transparent_pixel_color

Specifies the color for transparent pixels when exporting images to formats such as TIFF or GIF.

Data Type	string
Default Value	RGB222

ui_fonts.header

Specifies the header font used by various UI components.

Data Type	string
Default Value	Windows: “family:Times New Roman-face:bold-size:18” UNIX: “family:helvetica-face:bold-size:18”

ui_fonts.large

Specifies the large font used by various UI components.

Data Type	string
Default Value	Windows: “family:MS Sans Serif-face:bold-size:12” UNIX: “family:courier-face:medium-size:14”

ui_fonts.medium

Specifies the medium font used by various UI components.

Data Type	string
Default Value	Windows: “family:MS Sans Serif-size:10” UNIX: “family:courier-face:medium-size:12”

ui_fonts.small

Specifies the small font used by various UI components.

Data Type	string
Default Value	Windows: “family:MS Sans Serif-size:8” UNIX: “family:courier-face:medium-size:10”

ui_fonts.tiny

Specifies the tiny font used by various UI components.

Data Type	string
Default Value	Windows: “family:MS Sans Serif-size:7” UNIX: “family:courier-face:medium-size:8”

ui_layout.hide_buttons

Specifies whether OPNET should hide action buttons. (This preference is useful for providing more workspace on small screens.) When TRUE, action buttons are hidden.

The value of this preference changes only at start-up.

Data Type	boolean
Default Value	FALSE

use_tooltip_as_link_title

Specifies whether OPNET should display the first line of the tooltip text as a label for links (subject to the setting of network_visualization.show_link_labels). When TRUE, tooltip text is used as a label. When FALSE, the link name is used as a label.

Data Type	boolean
Default Value	FALSE

use_startup_wizard

Specifies whether to use the startup wizard when specifying a new scenario.

Data Type	boolean
Default Value	FALSE

This preference is used with the preference suppress_use_startup_wizard_checkbox. Table 3-8 lists the effects of different settings for these options.

Table 3-8 Effects of use_startup_wizard and suppress_use_startup_wizard_checkbox

	suppress_use_startup_wizard_checkbox	
	TRUE	**FALSE**
use_startup_wizard= TRUE	The checkbox does not appear and the startup wizard is used by default.	The checkbox appears and the startup wizard is used if the checkbox is selected.
use_startup_wizard= FALSE	The checkbox does not appear and the startup wizard is not used.	The checkbox appears and the startup wizard is used if the checkbox is selected. Additionally, if the checkbox is selected, the setting for preference use_startup_wizard is set to TRUE.
End of Table 3-8		

verbose_report

Specifies that reports generated by the Generate Text Report command in the Process, Packet Format, and ICI Editors should be done in verbose mode. In non-verbose mode, only the name, type, value, and default value of an object attribute is printed.

Data Type	boolean
Default Value	FALSE

vudoc_prog

Specifies the path name of the Acrobat Reader program. Under UNIX, the value of this preference typically remains blank because the program is installed in a fixed location (sys/doc/viewer). Under Windows, Sentinel sets this value when it is installed.

Data Type	string
Default Value	""

win_height

Specifies the default height (in pixels) for tool windows that do not have a natural size. This preference will not affect tool windows whose positions are remembered if the ui.remember_window_sizes_and_positions preference is TRUE.

Data Type	integer
Default Value	600

win_width

Specifies the default width (in pixels) for tool windows that do not have a natural size. This preference will not affect tool windows whose positions are remembered if the ui.remember_window_sizes_and_positions preference is TRUE.

Data Type	integer
Default Value	760

win_x

Specifies the default x position (in pixels) for the first tool window. This preference will not affect tool windows whose positions are remembered if the ui.remember_window_sizes_and_positions preference is TRUE.

Data Type	integer
Default Value	40

win_x_inset

Specifies the minimum horizontal distance (in pixels) from the left or right screen edge that is reserved when a window is opened.

Data Type	integer
Default Value	0

win_x_stagger

Specifies the horizontal offset (in pixels) for subsequent tool windows after the first one is opened.

Data Type	integer
Default Value	15

win_y

Specifies the default y position (in pixels) for the first tool window. This preference will not affect tool windows whose positions are remembered if the ui.remember_window_sizes_and_positions preference is TRUE.

Data Type	integer
Default Value	40

win_y_inset

Specifies the minimum vertical distance (in pixels) from the top or bottom screen edge that is reserved when a window is opened.

Data Type	integer
Default Value	0

win_y_stagger

Specifies the vertical offset (in pixels) for subsequent tool windows after the first one is opened.

Data Type	integer
Default Value	35

window_icon

Specifies the name of the icon to use for application windows.

Data Type	string
Default Value	“opnet_logo_small”

wrap_world_links

Specifies whether links should visually extend more than 180 degrees in longitude. When TRUE, links are drawn using the shortest route between sites, even if this means wrapping from the left side of the screen to the right. When FALSE, all links are drawn continuously across the map.

Data Type	boolean
Default Value	TRUE

Licensing

Licensing preferences indicate options that affect the licensing operations of OPNET programs. These are described in the Licensing chapter of the *Administrator Guide*. One additional preference that can be used with the License Manager, tool_name, is a GUI preference, and is described in GUI below.

Printing

The printing preferences control the generation of print output files (both graphics and reports) and the spooling of print output files to a PostScript printer. Sentinel programs that have the capability of generating print output files accept these preferences.

printcmd

Specifies the printer spooling program that will be invoked with PostScript output files generated by graphical or report printing operations. The default value of this preference is lpr, the standard printer spooling program of the Berkeley versions of UNIX. Another typical value assigned to this preference is lp, the standard printer spooling program of the System V versions of UNIX. In addition to the printer spooling program name, the value of this preference can include command line flags that appear before the name of the file being spooled (for example, lpr -h).

Data Type	string
Default Value	"lpr"

printername

Specifies the default printer. This will typically be the name of the default printer specified in the UNIX environment.

Data Type	string
Default Value	""

print_graph_method

Specifies how visible graphs will be printed:

- monochrome—black and white, with symbols used to identify traces.
- white-background—grayscale or color, with a white background.
- full-color—grayscale or color, with the background color assigned to the graph.

Data Type	enum
Default Value	monochrome

print_page_limit

Specifies the maximum number of pages that will be printed in one print request.

Data Type	integer
Default Value	100

User Lock

The user lock preferences provide options that affect the interpretation of user model write locks by Sentinel programs. Sentinel programs that write out model files accept these preferences.

edit_lock_mode

Specifies that models currently in use will have an edit lock placed on the appropriate model files.

The value of this preference changes only at start-up.

Data Type	boolean
Default Value	FALSE

named_locks

Specifies that a development name is to be used as the owner name for user locks on model files. If this preference is set to TRUE, and the development name is not specified by the user_lock_devname preference, a prompt for the development name will appear in the parent shell of the affected program. If this preference is disabled, the user's account name is used as the owner of user locks.

The value of this preference changes only at start-up.

Data Type	boolean
Default Value	FALSE

user_lock_devname

Specifies the development name to be used as the owner name for user locks on model files. The value of this preference is ignored if the preference named_locks is FALSE.

The value of this preference changes only at start-up.

Data Type	boolean
Default Value	FALSE

Web Reporting

The web reporting preferences provide options that affect the creation of HTML-based reports by Sentinel.

web_rep_graph_enable

When TRUE, enables generation of web-compatible (GIF) graphics from analysis panels.

Data Type	boolean
Default Value	FALSE

web_rep_graph_height

Specifies the default height of a web-based report, in pixels.

Data Type	integer
Default Value	300

web_rep_graph_width

Specifies the default width of a web-based report, in pixels.

Data Type	integer
Default Value	500

web_rep_home_url

Specifies the URL of the simulation reports home page.

Data Type	string
Default Value	""

web_rep_storage_dir

Full path of directory used to store web-based report files.

Data Type	string
Default Value	""

web_rep_vendor_urls

Specifies a list of vendor links to be added to the web report home page, allowing you to customize your report home page or link to other web-based reports.

There must be exactly three consecutive lines in the following order for each vendor:

- Vendor name
- Vendor URL
- Vendor image name

Data Type	string
Default Value	""

XML

The XML preferences provide options that affect the import and export of XML files by OPNET programs.

xml_export_postproc_function

Specifies the name of a function to invoke for XML export post-processing.

Type	string
Default Value	“Nt_Xmle_Postproc_Default”

xml_export_postproc_library

Specifies the name of a library containing the XML export post-processing function.

Type	string
Default Value	“nt_xmle_postproc”

xml_export_postproc_verbose_attrs

Specifies whether OPNET should export all attributes, regardless of intended status. When TRUE, all attributes are exported.

Type	boolean
Default Value	FALSE

xml_import_postproc_function

Specifies the name of a function to invoke for XML import post-processing.

Type	string
Default Value	“Nt_Xmli_Postproc_Default”

xml_import_postproc_library

Specifies the name of a library containing the XML import post-processing function.

Type	string
Default Value	“nt_xmli_postproc”

4 Program Descriptions

This chapter describes how to operate the programs, shell scripts, and batch files that comprise SP Sentinel. It describes the external interface of each program and utility, including the preferences that the programs accept.

The main Sentinel program (spsentinel) and several others use a graphical user interface (GUI), which is described in the User Interface chapter of the *User Guide*. Detailed instructions for using spsentinel to build models and run simulations are given in the User Guide.

The following table lists the programs and utilities in the order in which they are discussed in this chapter.

Table 4-1 SP Sentinel Programs and Utilities

Name	Type	Description
op_clrtmp (U) op_clrtmp.bat (W)	shell script batch file	Removes all files from the temporary file directory of the user's administration directory (<HOME>/op_admin/tmp).
op_flse	program	Releases licenses to the License Manager when license-controlled programs are exited.
op_install	shell script	Provides a method for automating opnet installation.
op_license_manager	shell script	Opens opnet in the manage license mode.
op_license_server	program	Issues licenses to licensed applications.
op_license_util	program	Does management operations for license servers.
op_manfile	program	Does management operations related to the tracking and identification of SP Sentinel model files, including file location, locks, and header contents.
op_vuerr	program	Prints function stack traceback and other program error information.
spsentinel	GUI program	Permits editing and viewing SP Sentinel models, executing simulations, and analyzing simulation output results.
End of Table 4-1		

op_clrtmp

The op_clrtmp shell script and op_clrtmp.bat batch file remove all files from the temporary file directory in your administration directory (<HOME>/op_admin/tmp).

Description

Several SP Sentinel programs generate temporary files in the course of executing. These files are stored in the temporary file directory (<HOME>/op_admin/tmp), as described in the section Temporary File Directory (tmp) on page SSR-2-14. Some temporary files are removed as soon as the program that created them completes execution. Other temporary files remain in this directory until they are explicitly removed.

After many invocations of SP Sentinel programs or many print operations, the temporary file directory can grow large, containing many files. When this occurs, all directory files can be removed manually (with the UNIX command `rm *`, for example). However, this approach is risky because it does not verify that the user is actually in the proper directory. Thus, an incorrect assumption about which directory is current can lead to accidental removal of valuable files.

A more reliable way to remove the files in the temporary directory is provided by the `op_clrtmp` shell script (UNIX) and `op_clrtmp.bat` batch file (Windows). These utilities can be invoked from any directory and will remove all files from <HOME>/op_admin/tmp, without risk of removing files in other directories.

Preferences

The op_clrtmp and op_clrtmp.bat utilities use no preferences.

op_flse

The op_flse program releases licenses to op_license_server when an application is exited.

Description

The op_flse program is executed as a background process by all licensed applications. It is the component of the licensing system that is responsible for releasing licenses back to the License Manager. The op_flse program releases licenses and not the applications themselves because the applications may occasionally exit in uncontrolled circumstances, and so lock up a permit.

A license represents the right to use the application for one session. The status of licenses is stored within the license file, a data file located in

- `<primary_hard_drive>:\OPNET_license\license_file` (Windows platforms)
- `/opt/OPNET_license/license_file` (UNIX platforms)

See the Licensing chapter in the *Administrator Guide* for details on licenses.

When a licensed application exits, the corresponding op_flse process running in the background detects the application process's death and sends a message to op_license_server indicating that the application's license is now free. Upon receipt, op_license_server changes the status of the license, allowing it to be reissued.

The op_flse program is not intended for direct execution by users.

WARNING—Do not kill op_flse processes. If you do, the associated licenses will never be returned to op_license_server.

Preferences

The op_flse program uses the following preference sets:

- Standard preferences
- Diagnostics preferences

For details on the standard preferences set, see Standard Preferences on page SSR-3-16. For details on other sets, see Preference Sets on page SSR-3-29.

The op_flse program-specific preferences are not described here because they are used only by licensed applications.

op_install

Description

The op_install program provides a method for automating OPNET installation. op_install is a script that guides you through the installation process. It should be run from the installation CDs.

For detailed instructions, see the installation sheets that came with the CDs or the technical support area of the OPNET Technologies web site.

Preferences

None, because this is a C Shell script.

op_license_manager

Description

The op_license_manager script opens opnet in the manage license mode. When opnet opens in the manage license mode, it does not use a license, but offers limited functionality. You can do license management tasks, but no other operations. The mode is useful for a few reasons:

- You can do license management tasks without depriving others of a license.
- You can reliably delete a particular license. If you launch opnet and then open the License Manager, you may find that the license you have been allocated is the one you wanted to delete. You cannot delete an in-use license.

You can also open opnet in this mode by launching it with the manage_licenses preference.

Preferences

None, because this is a C Shell script.

op_license_server

Description

The op_license_server program runs as a service (on Windows platforms) or as a daemon process (on UNIX platforms), issuing licenses to licensed applications. A daemon process is one that executes continuously without being attached to a user shell.

op_license_server manages the allocation of licenses. See the Licensing chapter of the *Administrator Guide* for details.

op_license_util

Description

The op_license_util utility does management operations for license servers.

op_license_util supports the following single-operation modes. These are specified by invoking op_license_util with the corresponding preferences:

- Check the status of a server ports (`-port_check`)
- Kill a license server (`-license_server_kill`)

Figure 4-1 Example Usage of op_license_util

```
% op_license_util -port_check
The status of the ports on "titan"
usable for the license server:

    port_a: unavailable
    port_b: available
    port_c: available

% op_license_util -license_server_kill TRUE -license_server unreliable -license_port
port_b
```

Preferences

This program supports the following preference sets:

- Standard preferences
- Diagnostics preferences
- Licensing preferences
- Printing preferences

For details on the Standard preferences set, see Standard Preferences on page SSR-3-16. For details on other sets, see Preference Sets on page SSR-3-29.

For details on the Licensing preference set, see Licensing in the *Administrator Guide*.

This program also supports the following additional preferences.

license_server_kill

Specifies that the license server process be killed. This operation sends a message to the license server, causing it to log its status just before exiting. This operation prevents op_license_util from presenting an interactive user interface. Only a user account that has write permissions on the license file can do this operation. The values of the license_server and license_port preferences are used to determine the location of the license server process. If these preferences are not set, the license file is checked for the location of the license server.

This preference can be used only from the command line.

Type	boolean
Default Value	FALSE

port_check

Specifies that the status of the three UDP/IP ports potentially used by the licensing system be checked for availability on the local host computer. A list of the three ports and their statuses is printed on the standard output device. This operation prevents op_license_util from presenting an interactive user interface.

This preference can be used only from the command line.

Type	boolean
Default Value	FALSE

op_manfile

The op_manfile program does management operations related to the tracking and identification of SP Sentinel model files, including file location, locks, header contents, and model security.

Description

SP Sentinel model files have three portions: a file header, file comments, and model data. The header portion of a model file contains tracking and identification information for the file, such as the time at which the file was created. The comment portion contains user-entered text descriptions of the file's data contents and modification history. Model-derived object code files also contain embedded, non-modifiable file headers. The header portion contains the following information about the file:

- model type
- creation time and date
- name and release of the creating SP Sentinel application
- earliest release of the creating SP Sentinel application that is compatible with the file's format
- last modification time and date
- version number of the file (that is, the number of times the file has been written to disk)
- user account name and host computer name associated with the last modification
- status of the file's edit lock (edit locks are automatically set by SP Sentinel applications that are editing a model file, to prevent concurrent editing of the same file)
- status of the file's user lock (user locks are manually set by users to indicate temporary ownership and prevent other users from modifying the model during a development phase)
- model security access information (status, number of registered sites, access level of the local site, and world access status)
- model suite information (suite name and description, often including the model vendor's name and contact information)

The header information can be useful in establishing the background of a model file. SP Sentinel applications that manipulate model files can modify the above information, but the only way to view this information is using op_manfile.

The op_manfile program provides the following operations, each of which must be specified from the command line. Only one operation can be done per op_manfile session.

Table 4-2 op_manfile Operations

Description	Preference
Find one or more models based on the file base name	-find
Print the file header	-header_print
Repair the file header	-header_repair
Update the file header	-header_update
Set the file version number	-version_set
Add comments to a file	-comments_add
Delete comments from a file	-comments_del
Print file comments	-comments_print
Set the edit lock	-edit_lock
Release the edit lock	-edit_unlock
Set the user lock	-user_lock
Release the user lock	-user_unlock
Display the file's checksum	-checksum
Register a model for access	-register
End of Table 4-2	

All op_manfile operations require a least one target model file, to which they are applied (some operations can support multiple target files). The target model file of op_manfile is specified by the preference m, mf, or f. Some commands (such as register) also take an optional source model file, which is specified by the preferences from, from_f, or from_mf. The general format of an op_manfile command is as follows:

```
op_manfile <operation> <target> <source>
```

Figure 4-2 shows an example of how to use the op_manfile program to register the suite of models of which oscar_mode_a is a member.

Figure 4-2 Example Usage of op_manfile (1)

```
op_manfile -register -suite -m oscar_mode_a

Send the following transaction code to the model vendor:
Transaction code> 0EDF.98DB.6438.F41A

Enter the approval code sent by the model vendor:
Approval code> 6833.FA47.17C1.BA1A.7C75.C9DC

--- op_manfile issued the following command: ---
/bin/cp /dtaylor/op_models/oscar_mode_a.nd.m \
/dtaylor/op_admin/bk/oscar_mode_a.nd.m

Success.
```

The next example illustrates using the op_manfile program to print out header information for a node model named atm16_crossconn.

Figure 4-3 Example Usage of op_manfile (2)

```
C: op_manfile -m atm16_crossconn -header_print

|-------------------------------------------------------------------------|
|file path        : C:\OPNET\10.5.A\models\std\atm\atm16_crossconn.nd.d    |
|-------------------------------------------------------------------------|
|file type        : Derived Node Model                                    |
|-------------------------------------------------------------------------|
|creation                                                                 |
|    date         : Wed Feb 28 14:11:09 1996                              |
|    application  : op_cvdlb2                                             |
|    release      : 10.0.A PL2 (build 2308)                               |
|    file data v. : 8.5.A                                                 |
|-------------------------------------------------------------------------|
|last modification                                                        |
|    date         : Fri Oct 03 22:26:29 2003                              |
|    version      : 27                                                    |
|    user         : opnet                                                 |
|    host         : sunblade60                                            |
|-------------------------------------------------------------------------|
|edit lock                                                                |
|    status       : unlocked                                              |
|-------------------------------------------------------------------------|
|user lock                                                                |
|    status       : unlocked                                              |
|-------------------------------------------------------------------------|
|security                                                                 |
|    status       : not protected                                         |
|-------------------------------------------------------------------------|
|suite info       : (none available)                                      |
|-------------------------------------------------------------------------|

C:
```

Preferences

The op_manfile program uses the following preference sets:

- Standard preferences
- Development preferences—core_dump, mem_track, mem_track_object, prof_track, prof_track_func, and prof_track_recurse only
- Diagnostics preferences

- File preferences—backup_max_count only
- Licensing preferences
- User Lock preferences

For details on the standard preferences set, see Standard Preferences on page SSR-3-16. For details on the Licensing preference set, see Licensing in the *Administrator Guide*. For details on other sets, see Preference Sets on page SSR-3-29.

The op_manfile program uses the following program-specific preferences.

all

Specifies that the selected operation be expanded in its scope, in an operation-specific manner. This preference modifies the following operations: edit_lock, edit_unlock, find, header_print, user_lock, and user_unlock.

For a lock-related operation, the all preference causes the operation to be applied to all model files located in the model directories (thus, specific files do not need to be specified by the f, m, or mf preferences).

For the find operation, the all preference causes all models with duplicate names in different model directories to be found. (Normally, a model found in a higher-precedence model directory will override and effectively hide duplicate models found in lower-precedence model directories).

For the print_header operation, the all preference causes the printing of multiple headers present within a file, instead of just the first header in the file. (Only simulations can contain multiple headers, due to the headers associated with each model-derived object code module in the simulation file).

Type	boolean
Default Value	FALSE

app_name

Specifies the name of the application identified in the target file header as the creator of the file. This preference's value is used by op_manfile when a file header is created with the header_add preference or when a file header is updated with the header_update preference.

Type	string
Default Value	""

checksum

Specifies that the checksum numbers of the target file be computed and printed. This operation can be applied to any type of file via the f preference, including program executables. The output of the operation is two integer checksum numbers, identified as Add and Xor because of their respective computation methods. File checksum numbers are used to determine whether a release file has been corrupted. Contact Technical Support to compare your checksums with the expected values.

Type	boolean
Default Value	FALSE

comments_add

Specifies that one comment be added to the target file. You enter the text of the comment after a prompt that appears on the standard output device. Comments are limited to one line of text; pressing the <Return> key completes the comment. Comments are tagged with the date, user account name, and file version associated with their addition to the file.

Type	boolean
Default Value	FALSE

comments_del

Specifies that all comments in the target file be deleted.

Type	boolean
Default Value	FALSE

comments_print

Specifies that all comments contained in the target file be printed to the standard output device. Each comment is separated by a line of underscore, and is tagged with the date, user account name, and file version number associated with the comment's addition to the file.

Type	boolean
Default Value	FALSE

edit_lock

Specifies that the target file's edit lock be set, as if the file were being edited by the program. This tags the file as "read only" and prevents all SP Sentinel programs from modifying it. This operation is not typically used; instead, you can temporarily lock a file with the user_lock preference.

Type	boolean
Default Value	FALSE

edit_unlock

Specifies that the target file's edit lock be reset, thus allowing modification by SP Sentinel programs. If the program exits abnormally, files that were being edited by the program may retain edit locks. This situation can be rectified with the edit_unlock preference.

Type	boolean
Default Value	FALSE

f

Specifies the full name and path of the target file on which operations will be done. The file names must include file type suffixes and absolute or relative paths.

Type	string
Default Value	""

find

Specifies that op_manfile search the model directories for the target file(s) and print information about their locations on the standard output device. The location information includes the target file names (with suffix), types, and model directories.

This operation does not request the selection of one target file if multiple model files match a base name specified by the m preference; it locates all model files that satisfy the target name. This operation cannot be used with the f preference.

Type	boolean
Default Value	FALSE

Specifies the base name of the source model from which to copy protection information (that is, the model name without the file suffix or path). The op_manfile program locates the specified model file in the model directories. If more than one model file is found with this base name and the specified operation requires one source file, op_manfile will request the selection of one file from a menu of options printed on the standard output device.

Type	string
Default Value	“”

from_f

Specifies the full name and path of the source model from which to copy protection information. The file name must include file suffix and absolute or relative path.

Type	string
Default Value	“”

from_mf

Specifies the full name and file suffix of the source model from which to copy protection information. The file name should not include a path (the specified model file will be located in the model directories).

Type	string
Default Value	“”

header_add

Specifies that header information be added to the target file.

Type	boolean
Default Value	FALSE

header_print

Specifies that the header information of the target file be printed on the standard output device. The header includes information about the file path and type, creation, last modification, and security.

Type	boolean
Default Value	FALSE

header_repair

Specifies that the header information of the target file be repaired. If the program issues an error message indicating that a model file is corrupted, this operation may be able to repair the damage if it is limited to the header.

Type	boolean
Default Value	FALSE

header_update

Specifies that the header information of the target file be updated. An update modifies all “last modification” fields in the header, including date, version number, user account name, and host computer name. The creating application name is also modified if the app_name preference is set.

Type	boolean
Default Value	FALSE

m

Specifies the base name of the target files on which operations will be done (that is, the model name without the file type suffix or path). The op_manfile program locates the specified model file in the model directories. If more than one model file is found with this base name and the specified operation requires one target file, op_manfile will request the selection of one target file from a menu of options printed on the standard output device.

Type	string
Default Value	“”

mf

Specifies the full name and file suffix of the target file on which operations will be done. The file name should not include a path (the specified model file will be located in the model directories).

Type	string
Default Value	“”

register

Registers a protected model file. Registration allows access to the model by users at a specified site (SP Sentinel installation).

Type	boolean
Default Value	FALSE

suite

Specifies that all model files in the same suite as the specified model should be registered. The suite preference must be used with the register preference.

Type	boolean
Default Value	FALSE

user_lock

Specifies that the target file’s user lock be set. This tags the file as belonging to the user account that set the lock and prevents SP Sentinel programs being run by other user accounts from modifying the file.

Type	boolean
Default Value	FALSE

user_unlock

Specifies that the target file’s user lock be reset, thus allowing programs run by any user account to modify it.

Type	boolean
Default Value	FALSE

version_set

Specifies that the target file's version number be set to the specified integer value (the default value of -1 specifies that no change be made).

Type	integer
Default Value	-1

op_vuerr

The op_vuerr program prints out function stack traceback and other program error information. The error information is extracted from an error log maintained by programs supplied by OPNET or based on OPNET object code libraries.

Description

The op_vuerr program provides a direct way to view the most recently logged errors from SP Sentinel programs. It opens up the error log, searches out the last error (or last *n* errors, if specified), and prints out the traceback information on the standard output device. The program can be executed from any directory.

Typically, SP Sentinel programs handle error conditions and program crashes using a unified error package. This package indicates the error by updating a message area (if the program has a graphical user interface) or by printing messages on the standard output device. In all cases, errors are also logged in an ASCII file named `err_log` within the administration directory (<*HOME*>/`op_admin`).

Each entry in `err_log` has the following components:

- Error classification, which indicates the severity of the error. The possible classifications are:
  - — Warning—A situation that may lead to problems and that should be checked by the user.
  - — Recoverable Error—A problem that does not prohibit the program from proceeding, but that may result in cancellation of the current operation.
  - — Program Abort—A problem that prohibits the program from reliably proceeding and therefore causes the program to immediately abort.
  - — Program Fault—A problem that causes the operating system to immediately abort the program in an uncontrolled manner.
- Date and time when the error occurred.
- Program in which the error occurred.
- Package of functions in which the error occurred.
- Function in which the error occurred.
- Description of the error, including variable values and related information which can serve to identify the error's cause.
- Contents of the program-maintained function stack, which can be used by technical support personnel to help identify the location and cause of the error.

The error log continues to grow as program errors occur and traceback information is added to the file. The error log is cumulative so as to provide a reliable description of a recent sequence of errors that users or technical support personnel can use to investigate the cause of the errors.

Typically, the error log will grow to many thousands of lines after a month of heavy use of SP Sentinel programs, so you should check the log's size regularly if disk space is at a premium. The error log file can be removed at any time to reduce disk space usage; it will be recreated as needed when the next error is logged.

Figure 4-4 Example Usage of op_vuerr

```
op_vuerr

<<< Program Abort >>>
* Time:      Fri May 17 17:31:48 1999
* Program:   op_runsim (Version 7.0.B)
* Package:   Simulation Kernel
* Function:  sim_err_print (severity_level, str0, str1)
* Error:     Error encountered rebuilding repository -- unable to proceed
T (0), EV (-1), MOD (NONE), PROC (sim_load_repos_load)

* Function call stack: (builds down)
------------------------------------------------------------
Call   Block
Count  Line#  Function

    ------------------------------------------------------------
0)          1    267  main (argc, argv, envp)
1)          1    204  sim_init (argc, argv, envp)
2)          1    148  sim_load_models (net_name)
3)          1    571  sim_load_repos_load (net_name)
4)          2    171  sim_err_print (severity_level, str0, str1)
5)          3    224  Vos_Error_Print (level, package, error0, error1, error2)
------------------------------------------------------------
```

Each line in the function call stack contains a call count and a block line number in addition to the function name and arguments. The call count specifies the total number of times that the function has been called since the start of program execution. The block line number specifies the last source-code line number

which was known to have executed within that function at the time of the error. The block line number is not a fine-grained measure and does not necessarily list the exact line number executed within the function at the time of the error. The example in Figure 4-5 shows this.

Figure 4-5 Sample Code with Four Error Locations

```
1   int function()
2       {
3       int x, y, z;
4           int *ptr;
5
6                z = 0;                          A
7                   for (x = 0; x < 10; x++)
8       {
9           for (y = 0; y < 10; y++)             B
10          {
11              z += y;
12              z += x;                         C
13          }
14      }
15          return z;                            D
16      }
```

Preferences

The op_vuerr program uses the following preference sets:

- Standard preferences
- Development preferences—core_dump, mem_track, mem_track_object, prof_track, prof_track_func, and prof_track_recurse only
- Diagnostics preferences—beep_count_confirm and beep_count_error only

For details on the standard preferences set, see Standard Preferences on page SSR-3-16. For details on other sets, see Preference Sets on page SSR-3-29.

The op_vuerr program uses the following program-specific preference.

num_err

Specifies the number of errors to be extracted from the error log and printed to the standard output device. The most recent *n* errors are printed by op_vuerr (the last *n* entries in the error log), where *n* is the value of this preference.

Type	integer
Default Value	1

spsentinel

The spsentinel program is a GUI-based, license-controlled application used to edit and view SP Sentinel models, execute simulations, and analyze simulation output results.

Description

This is the main program of SP Sentinel. It is here that you use the Project Editor to conduct modeling and simulation projects.

The spsentinel program supports a large array of preferences that affect its operation, as described in the following subsection. Detailed operational instructions for Sentinel are presented throughout the *User Guide*. An introduction to using Sentinel is presented in the online tutorial.

Preferences

The spsentinel program uses the following preference sets:

- Standard preferences
- Development preferences
- Diagnostics preferences
- File preferences
- GUI preferences
- Licensing preferences
- Printing preferences
- User Lock preferences
- Web Reporting preferences
- XML preferences

For details on the Standard preferences set, see Standard Preferences on page SSR-3-16. For details on the Licensing preference set, see Licensing in the *Administrator Guide*. For details on the other sets, see Preference Sets on page SSR-3-29.

This program also supports preferences for specific add-on modules; these preferences are described in the corresponding module manuals.

The spsentinel program uses the following program-specific preferences.

aliases_verify

When set to TRUE, spsentinel verifies that all node aliases in the current scenario are unique. spsentinel does this whenever you change a node alias, import a topology or import an alias set. A text edit pad lists any duplicate aliases found.

Type	boolean
Default Value	TRUE

an_mono_sym

Specifies that analysis panels drawn on monochrome displays should visually distinguish separate traces with unique marker symbols (such as triangles and circles). The value of this preference also applies to multi-trace panels that are printed, because print output is monochrome.

Type	boolean
Default Value	TRUE

apply_all_aliases

Specifies the persistence of aliases assigned to network objects in the Unrecognized Traffic Assistant after traffic import. By default, node aliases are persistent program-wide. Setting this preference to FALSE makes node aliases persistent project-wide.

Type	boolean
Default Value	TRUE

auto_fire

When TRUE, specifies that analysis panels should automatically display their traces. If this preference is set to FALSE, trace drawing must be manually triggered by selecting the Show Statistic checkbox in the Edit Panel dialog box.

Type	boolean
Default Value	TRUE

bgutil_errlog_incident_limit

Specifies the incident limit for the link load import error logs.

Type	integer
Default Value	10

bgutil_gbu_bucket_size

Specifies the bucket size to be used for ASCII (`.gbu`, `.gbu2`) imports. This preference can be set also from the Import Link Baseline Loads from ASCII FIles dialog box.

Type	string
Default Value	“As Is”

bgutil_gbu_bucket_type

Specifies the bucket mode to be used for ASCII (`.gbu`, `.gbu2`) imports. This preference can be set also from the Import Link Baseline Loads from ASCII FIles dialog box.

Type	string
Default Value	“Average”

bgutil_gbu_file_list

Specifies a list of files to be imported as ASCII (`.gbu`, `.gbu2`) files. This preference can be set also from the Import Link Baseline Loads from ASCII FIles dialog box.

Type	string list
Default Value	<empty>

bgutil_gbu_from_time

Specifies the start of the time window for filtering ASCII (`.gbu`, `.gbu2`) data. This preference can be set also from the Import Link Baseline Loads from ASCII FIles dialog box.

Type	string
Default Value	“ALL VALUES”

bgutil_gbu_overwrite

Specifies whether ASCII (`.gbu`, `.gbu2`) imports should overwrite existing data. When TRUE, existing data is overwritten. This preference can be set also from the Import Link Baseline Loads from ASCII FIles dialog box.

Type	boolean
Default Value	TRUE

bgutil_gbu_to_time

Specifies the end of the time window for filtering ASCII (`.gbu`, `.gbu2`) data. This preference can be set also from the Import Link Baseline Loads from ASCII FIles dialog box.

Type	string
Default Value	“ALL VALUES”

bgutil_ignore_alias_case

Specifies whether to ignore the case of alias names when importing background utilization (link loads). When TRUE, case is ignored.

Type	boolean
Default Value	TRUE

comp_target_type.development_parallel_32

Specifies whether to generate this type of object file when compiling. Together with the other comp_target_type preferences, this allows you to produce multiple targets from a single source.

The value of this preference changes only at start-up.

Data Type	boolean
Default Value	FALSE

comp_target_type.development_parallel_64

Specifies whether to generate this type of object file when compiling. Together with the other comp_target_type preferences, this allows you to produce multiple targets from a single source.

The value of this preference changes only at start-up.

Data Type	boolean
Default Value	FALSE

comp_target_type.development_sequential_32

Specifies whether to generate this type of object file when compiling. Together with the other comp_target_type preferences, this allows you to produce multiple targets from a single source.

The value of this preference changes only at start-up.

Data Type	boolean
Default Value	TRUE

comp_target_type.development_sequential_64

Specifies whether to generate this type of object file when compiling. Together with the other comp_target_type preferences, this allows you to produce multiple targets from a single source.

The value of this preference changes only at start-up.

Data Type	boolean
Default Value	TRUE

comp_target_type.optimized_parallel_32

Specifies whether to generate this type of object file when compiling. Together with the other comp_target_type preferences, this allows you to produce multiple targets from a single source.

The value of this preference changes only at start-up.

Data Type	boolean
Default Value	FALSE

comp_target_type.optimized_parallel_64

Specifies whether to generate this type of object file when compiling. Together with the other comp_target_type preferences, this allows you to produce multiple targets from a single source.

The value of this preference changes only at start-up.

Data Type	boolean
Default Value	FALSE

comp_target_type.optimized_sequential_32

Specifies whether to generate this type of object file when compiling. Together with the other comp_target_type preferences, this allows you to produce multiple targets from a single source.

The value of this preference changes only at start-up.

Data Type	boolean
Default Value	FALSE

comp_target_type.optimized_sequential_64

Specifies whether to generate this type of object file when compiling. Together with the other comp_target_type preferences, this allows you to produce multiple targets from a single source.

The value of this preference changes only at start-up.

Data Type	boolean
Default Value	FALSE

disk_cleanup.clean_automatically

Specifies whether OPNET should automatically maintain the files it creates on disk. When TRUE, “old” files are automatically removed from disk as specified by the other disk_cleanup preferences. This preference can be changed also with the File > Delete Temporary Files… operation.

Data Type	boolean
Default Value	TRUE

disk_cleanup.clean_duplicate_scenarios

Specifies whether OPNET should automatically maintain the number of automation-generated scenarios in a project. When TRUE, duplicate scenarios older than specified by disk_cleanup.clean_duplicate_scenarios_days are automatically removed from disk when disk cleanup is executed. This preference can be changed also with the File > Delete Temporary Files… operation.

Data Type	boolean
Default Value	TRUE

disk_cleanup.clean_duplicate_scenarios_days

Specifies the number of days that automation-generated scenarios will be kept before being automatically removed from disk. This preference is used only when disk_cleanup.clean_duplicate_scenarios is TRUE. Its value can be changed also with the File > Delete Temporary Files… operation.

Data Type	integer
Default Value	30

disk_cleanup.clean_error_log

Specifies whether OPNET should automatically maintain the size of the error log. When TRUE, error log entries older than specified by disk_cleanup.clean_error_log_days are automatically removed from disk when disk cleanup is executed. This preference can be changed also with the File > Delete Temporary Files… operation.

Data Type	boolean
Default Value	TRUE

disk_cleanup.clean_error_log_days

Specifies the number of days that error log entries will be kept before being automatically removed from the log. This preference is used only when disk_cleanup.clean_error_log is TRUE. Its value can be changed also with the File > Delete Temporary Files… operation.

Data Type	integer
Default Value	30

disk_cleanup.clean_session_log

Specifies whether OPNET should automatically maintain the size of the session log. When TRUE, session log entries older than specified by disk_cleanup.clean_session_log_days are automatically removed from disk when disk cleanup is executed. This preference can be changed also with the File > Delete Temporary Files… operation.

Data Type	boolean
Default Value	TRUE

disk_cleanup.clean_session_log_days

Specifies the number of days that session log entries will be kept before being automatically removed from the log. This preference is used only when disk_cleanup.clean_session_log is TRUE. Its value can be changed also with the File > Delete Temporary Files… operation.

Data Type	integer
Default Value	30

disk_cleanup.clean_temporary_files

Specifies whether OPNET should automatically maintain the temporary files in `<HOME>\op_admin\tmp`. When TRUE, temporary files older than specified by disk_cleanup.clean_temporary_files_days are automatically removed from disk. This preference can be changed also with the File > Delete Temporary Files… operation.

Data Type	boolean
Default Value	TRUE

disk_cleanup.clean_temporary_files_days

Specifies the number of days that temporary files will be kept before being automatically removed from disk. This preference is used only when disk_cleanup.clean_temporary_files is TRUE. Its value can be changed also with the File > Delete Temporary Files… operation.

Data Type	integer
Default Value	7

edit_attr_undo_warning_prompt

If TRUE, specifies that OPNET should display a warning dialog for non-undoable attribute-edit operations.

Type	boolean
Default Value	TRUE

ets_options

Specifies a list of unlicensed ETS-based product options.

The value of this preference changes only at start-up.

Type	string list
Default Value	<empty>

event_speed_parameter

This preference provides an advanced event management system that can be tuned to maintain peak performance when simulating very large systems. Larger values will result in faster simulations. The trade-off is that higher values of event_speed_parameter require additional memory.

Optimal values depend on the type and size of models. You may want to experiment with different values and compare the simulation's performance. In general, increasing this parameter will not yield significant performance gains for small, lightly-loaded systems.

Type	integer
Default Value	100,000

file_abort_read_suppress

Prevents a program abort when spsentinel reads a corrupted node model.

Type	boolean
Default Value	FALSE

flow_import_errlog_incident_limit

Specifies the maximum number of log messages recorded during flow imports for a particular category, class, and subclass.

Type	integer
Default Value	100

hpov_default_server

Specifies the default HP OpenView network topology server to be used for imports.

Type	string
Default Value	“”

import_default_interface_type

Specifies the default interface type assigned during a topology import if a port or link’s interface type is not specified or cannot be identified. The interface type’s protocol and (optionally) speed can be specified by this preference (for example, ethernet:10BaseT is a valid default interface type).

Type	string
Default Value	empty
Constraints	Must be a valid protocol type and speed, if appropriate

import_default_machine_type

Specifies the default machine type assigned during a topology import if an object’s machine type is not specified or cannot be identified (for example, switch is a valid default machine type).

Type	string
Default Value	empty
Constraints	Must be a valid machine type

import_mem_stats_log

Specifies whether OPNET logs memory usage statistics during topology import operations. When FALSE, memory statistics are not logged.

Type	boolean
Default Value	FALSE

ip_address_change_notify_limit

Specifies the maximum number of change messages displayed when auto-assigning or clearing IP addresses.

Type	integer
Default Value	0

javascript_enable

Specifies whether OPNET can use Javascript in the creation of results web reports. When TRUE, Javascript can be used.

Type	boolean
Default Value	TRUE

ma_errlog_incident_limit

Specifies the maximum number of incidents to be reported in a model assistant error log.

Type	integer
Default Value	10

manage_licenses

Specifies that spsentinel start with the License Manager running. In this mode, spsentinel does not use a license, opens the License Manager automatically, and allows only limited functionality. This mode is useful when you are doing license management functions only. You must use this mode if you are deleting all licenses, because the License Manager will not allow you to delete a license that is in use.

Type	boolean
Default Value	FALSE

model_assistant_adjust_view_after_apply

Specifies whether autozoom is applied to the workspace display following application of a model assistant file. When TRUE, the display is automatically zoomed to the affected objects.

The value of this preference changes only at start-up.

Type	boolean
Default Value	FALSE

model_assistant_configuration_file

Specifies a configuration file for the model assistant.

The value of this preference changes only at start-up.

Type	string
Default Value	“”

net_report_link_detail_attrs

Specifies which link model preferences should be included in the link details section of the network summary report. If this preference and net_report_link_tooltip_alias are set to the default values, the link details section lists the name and link type of each link, organized by subnet. When modifying this preference, enter model attribute names exactly as shown in the Attributes dialog box.

Type	string list
Default Value	tooltip

net_report_link_summary_attrs

Specifies which link model attributes should be included in the link summary section of the network summary report. If this preference and net_report_link_tooltip_alias are set to the default values, the link summary section lists the total number of links and the number (count) by link type. When modifying this preference, enter model attribute names exactly as shown in the Attributes dialog box.

Type	string list
Default Value	tooltip

net_report_link_tooltip_alias

Specifies the tooltip preference used in link sections of the network summary report. This is different from the UI tooltip (the explanatory text in a yellow box that appears when you place the cursor over buttons).

Type	string list
Default Value	Link Type

net_report_node_detail_attrs

Specifies which node model attributes should be included in the node details section of the network summary report. If this preference and net_report_node_tooltip_alias are set to the default values, the node details section lists the name and device type of each node, organized by subnet. When modifying this preference, enter model attribute names exactly as shown in the Attributes dialog box.

Type	string list
Default Value	tooltip

net_report_node_summary_attrs

Specifies which node model attributes should be included in the node summary section of the network summary report. If this preference and net_report_node_tooltip_alias are set to the default values, the node summary section lists the total number of nodes and the number (count) by device type. When modifying this preference, enter model attribute names exactly as shown in the Attributes dialog box.

Type	string list
Default Value	tooltip

net_report_node_tooltip_alias

Specifies the tooltip preference used in node sections of the network summary report. This is different from the UI tooltip (the explanatory text in a yellow box that appears when you place the cursor over buttons).

Type	string list
Default Value	Device Type

network_diff.default_live_spec

Specifies the default specification file for Live Object/Attribute Difference behavior.

Type	string
Default Value	“op_live_diff”

network_diff.default_report_spec

Specifies the default specification file for Object/Attribute Difference reports.

Type	string
Default Value	“op_network_diff”

network_diff.output_dir

Specifies the directory to which Object/Attribute Difference report files will be written.

Type	string
Default Value	“”

network_diff.suppress_live_legend

Specifies whether OPNET should suppress the legend that appears in the lower left-hand corner of the workspace while Live Object/Attribute Differences is active. This legend describes the active tracking mode and the meaning of each of the icons used to mark the various network objects. When FALSE, the legend is shown. When TRUE, the legend is suppressed.

Type	boolean
Default Value	FALSE

network_palette

Specifies a custom model list or set of keywords to be used initially with the object palette in the Project Editor. If this list is empty, no keywords or model list are set when SP Sentinel starts and the object palette displays icons for all models in your model directories. Specifying this preference can save time when you open the object palette, because the palette displays fewer models through which to search.

The value of this preference must be one of the following:

- One or more valid keywords. Multiple keywords must be separated by spaces.
- The reserved word custom_model_list followed by a space and the name of an existing custom model list.

You can specify keywords or a custom model list with this preference, but not both at once.

Type	string list
Default Value	“”
Constraints	see description

network_palette.fixed_subnet_icon

Specifies the icon used for fixed subnet objects in the object palette. You can specify an icon from any database listed in the “icon_dbs” line in your spsentinel.ets file. If this preference is an empty string, the fixed subnet object will not appear in the object palette.

Type	string
Default Value	“subnet”
Constraints	see description

network_palette.group_aggregation

Specifies the self-description core characteristic used to group objects in the object palette when model aggregation is enabled.

Type	string
Default Value	“System Object ID”

network_palette.keep_above

Specifies whether OPNET should prevent the object palette from being hidden behind the editor window. When TRUE, the object palette is kept in front of the editor window (but not other windows).

Type	boolean
Default Value	TRUE

network_palette.mobile_subnet_icon

Specifies the icon used for mobile subnet objects in the object palette. You can specify an icon from any database listed in the “icon_dbs” line in your spsentinel.ets file. If this preference is an empty string, the mobile subnet object will not appear in the object palette.

Type	string
Default Value	“subnet (mobile)”

network_palette.only_one

Specifies whether OPNET should limit a network model to only one open object palette at a time. When FALSE, any number of object palettes can be open for a network model. When TRUE, an attempt to open a second object palette will bring the existing object palette window to the front and activate it.

Type	boolean
Default Value	FALSE

network_palette.satellite_subnet_icon

Specifies the icon used for satellite subnet objects in the object palette. You can specify an icon from any database listed in the “icon_dbs” line in your spsentinel.ets file. If this preference is an empty string, the satellite subnet object will not appear in the object palette.

Type	string
Default Value	“subnet (mobile)”

nmx_no_dynamic

Specifies whether to allow use of DNS for HP NetMetrix traffic import. Setting this preference to TRUE forces NetMetrix to ignore DNS for resolution of addresses found in the archives being read.

Otherwise, if NetMetrix was installed to use DNS, importing archives can be very lengthy, as every Internet address found in the archive will be checked to find the corresponding host name. If the address is bogus or the corresponding site cannot be found, the search ends after a lengthy time-out, giving the impression that the import is going nowhere.

Type	boolean
Default Value	TRUE

nt_model_post_conversion_library

Specifies a library containing a post-conversion pre-processor function used to perform any additional cleanup work after automatic model conversion.

Type	string
Default Value	“”

nt_model_post_conversion_proc

Specifies a post-conversion pre-processor function used to perform any additional cleanup work after automatic model conversion.

Type	string
Default Value	“”

nt_model_pre_conversion_library

Specifies a library containing a pre-conversion pre-processor function used to perform any setup work for automatic model conversion.

Type	string
Default Value	“”

nt_model_pre_conversion_proc

Specifies a pre-conversion pre-processor function used to perform any setup work for automatic model conversion.

Type	string
Default Value	“”

num_collect_values

Specifies the default maximum number of values to collect for each statistic during a simulation. This value can be changed for a given scenario in the Configure/Run DES dialog box.

Type	integer
Default Value	100

parallel_sim.num_processors

Specifies the number of processors to be used by the Simulation Kernel. This preference applies only to the parallel kernels. If this preference is set to 0 or Maximum, the simulation will try to determine the number of CPUs available and use them all. This value can be changed for a given scenario in the Configure/Run DES dialog box.

WARNING—Requesting simulations to use more CPUs than are physically present and available will degrade simulation performance dramatically.

Type	integer
Default Value	1

product

Specifies an OPNET product, which determines the feature set of the opnet program. The possible product types are

- OPNET Development Kit (“odktools”)
- Modeler (“modeler”)
- SP Guru (“spguru”)
- IT Guru (“itguru”)
- SP Sentinel (“spsentinel”)
- IT Sentinel (“itsentinel”)

- CAPEX (“capex”)
- Netbiz (“netbiz”)
- WDM Guru (“wdmguru”)

The value of this preference changes only at start-up.

Type	enum
Default Value	spsentinel

product_options

Specifies a list of modules that add additional functionality to SP Sentinel.

Type	string
Default Value	“”
Constraints	must be a valid product option

repositories

Specifies the set of code repositories to load for simulation.

Type	string list
Default Value	“”

show_import_help

Specifies whether OPNET should display an import help dialog when failing to connect to a server. When TRUE, the help dialog is displayed.

Type	boolean
Default Value	TRUE

show_map_edit_mode_help_msg

Specifies whether OPNET should display a help message when entering map edit mode in the Project Editor. When TRUE, the help message is displayed.

Type	boolean
Default Value	TRUE

spreadsheet_prog

Specifies the path name of the spreadsheet program that launches when data or a report summary is exported. The value can be simply the program name if the program is in a directory specified by the path variable.

If no value is specified, the export operation still takes place, but the spreadsheet program does not launch. The data is exported to a text file stored in the op_admin/tmp directory.

The value of this preference changes only at start-up.

Type	string
Default Value	“”

static_bk_util.archive_dir

Specifies a list of directories in which background utilization (link load) data files are stored. SP Sentinel looks in this directory when you import data from a NetworkHealth (.dci) data file, an MRTG configuration (.cfg) file, or a General Background Utilization (.gbu) file.

Type	string
Default Value	“”

top_n_results_min_significant_digits

Specifies the minimum number of significant digits in a Find Top Results report.

Type	integer
Default Value	0

traffic_archive_dir

Specifies the path to a directory in which HP NetMetrix archive files can be found.

Type	string
Default Value	“”

traffic_default_interval

Specifies the interval (in seconds) used to subdivide imported traffic when intervals are not otherwise known.

Type	double
Default Value	30
Units	seconds

traffic_import_suppress_unrecognized_sources_dialog

Specifies whether OPNET displays the unrecognized sources dialog box after a traffic flow import. When FALSE, the dialog box is displayed; when TRUE, it is suppressed.

Type	boolean
Default Value	FALSE

traffic_preprocessor_prog

Specifies a list of external preprocessing scripts used to convert ASCII traffic files to Flexible CSV (.tr2) format. These scripts appear in the Import Traffic dialog box’s Traffic Preprocessor menu.

Note—You must include the full path name for any scripts located outside your user path.

Type	string list
Default Value	“”

traffic_servers

Specifies a list of available servers (that is, hostnames) on which op_servd and op_rtid are installed. Through these programs, OPNET can access the traffic files on remote machines (as specified by `traffic_archive_dir` in the op_rtid.ef environment file on the remote machine).

Type	string
Default Value	“”

undo_stack_size

Specifies the maximum number of undoable operations.

Type	integer
Default Value	10

use_metric

Specifies whether the Startup Wizard should use metric units as a default.

Type	boolean
Default Value	TRUE

use_network_editor_warning

Specifies whether OPNET should display migration information when starting the Network Editor.

Type	boolean
Default Value	TRUE

use_scenario_prompt

Specifies that SP Sentinel should prompt for a scenario name when creating a new project.

Type	boolean
Default Value	TRUE

use_startup_wizard

Specifies that the Startup Wizard will launch automatically each time a new scenario is created.

You can also control the Startup Wizard via the Run Startup Wizard item in the Tools menu and the “Use Startup Wizard for each new scenario” checkbox in the initial Startup Wizard dialog box. These methods supersede this preference: unchecking the checkbox will prevent the Startup Wizard from launching even if this preference is set to TRUE, and selecting the menu item will launch the Startup Wizard even if this preference is set to FALSE.

Type	boolean
Default Value	TRUE

value_notch_filter_tolerance

Specifies the range used by the value_notch filter to generate an output statistic. Given an input statistic T_0 and an ordinate value x, the value_notch filter eliminates T_0 if it is approximately equal to x. “Approximately equal” is defined as being greater than x – value_notch_filter_tolerance and less than x + value_notch_filter_tolerance. Otherwise the value_notch filter simply sends T0 to the output statisticTout.

Type	boolean
Default Value	TRUE

word_processing_prog

Specifies the path name of the word processing program that launches when a text report is generated. The value can be simply the program name if the program is in a directory specified by the path variable.

If no value is specified, the report is still generated, but the word processing program does not launch. The report is stored in the `op_admin/tmp` directory.

The value of this preference changes only at start-up.

Type	string
Default Value	“”

5 Icon Database Editor

The icons you see in the spsentinel interface are organized into icon database files, with the suffix .icons. Many icon databases are included as part of spsentinel; you can also create your own icon databases.

You can modify icon databases; you can also modify icons. Use the Icon Database Editor to modify the icon database. Use the Image Editor to modify individual icons. You cannot create one icon file, but you can create an icon database file that contains one icon.

Note—You cannot save an individual icon while in the Image Editor. To write the data related to an individual icon to disk, close the Image Editor and choose the Save operation in the Icon Database Editor to save the entire database.

The icon databases included with spsentinel are stored in several locations. For example, the database Cisco.icons (which has icons that represent Cisco products) is in `<reldir>/models/vendor_models/Cisco_Systems`. Other icon databases are in the appropriate model directory or in the sys directory. When you create an icon database, it is stored in your default directory (the first directory listed in your mod_dirs preference, typically op_models).

The table below lists common tasks related to icon databases and icons, with cross-references to other parts of this manual.

Table 5-1 Common Icon Tasks

Task	Reference
Create a new icon database	Icon Database Editor Menus on page SSR-5-2
Cut, copy, paste, or delete an icon in an icon database	Icon Database Editor Menus on page SSR-5-2
Create a new icon	Icon Database Editor Menus on page SSR-5-2
Edit an existing icon	Image Editor on page SSR-5-3
End of Table 5-1	

Icon Database Editor Menus

The Icon Database Editor provides operations for managing icons. You access these operations from the menu bar, which contains the following menus:

- File—contains operations that relate to high-level functions such as opening, closing, and saving icon databases. These operations are summarized in section the User Interface chapter of the SP Sentinel *User Guide*.
- Edit—contains operations that allow you to cut, copy, or paste individual icons in the icon database. The Edit menu also includes two operations that are specific to the Icon Database Editor:
  - New Icon adds a new icon of the default design to the database. You can then edit and rename this template icon in the Image Editor.
  - Delete Icon deletes the icon from the database.
- Windows—lists all open editor windows and allows you to make one active, as described in the User Interface chapter of the SP Sentinel *User Guide*.
- Help—provides access to context-sensitive help, product documentation and tutorial, the License Manager, and other options, as summarized in the User Interface chapter of the SP Sentinel *User Guide*.

Right-clicking on an icon in the database brings up the Icon pop-up menu, which provides access to the following operations:

Table 5-2 Icon Pop-Up Menu Operations

Menu item	Description
Edit	Opens the Image Editor with the selected icon
Set Name	Allows you to enter or edit the name of the icon
Delete	Deletes the icon
Cut	Removes the icon from the workspaces and places it on the clipboard
Copy	Places a copy of the icon on the clipboard
End of Table 5-2	

Right-clicking on the workspace brings up the Workspace pop-up menu, which provides access to the New Icon or Paste operations.

Image Editor

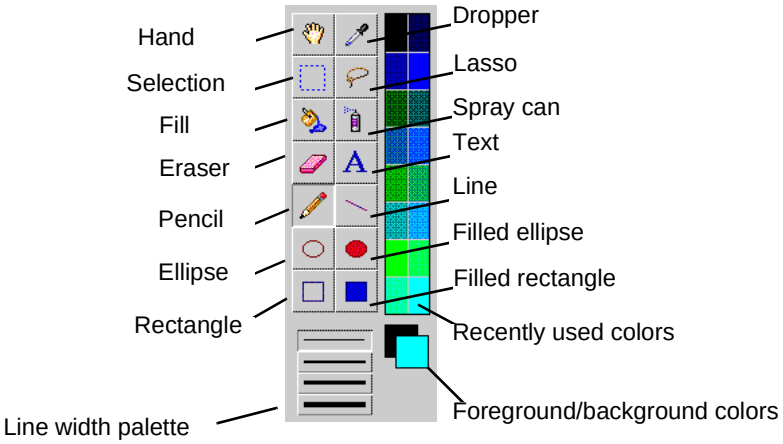
With the Image Editor, you can import, edit, and export icons. You access these operations from the menu bar, which contains the following menus:

- File—contains operations that relate to high-level functions such as closing, importing, exporting, and saving icons. These operations are summarized in section File Menu on page SSR-5-6.
- Edit—contains operations that allow you to edit the appearance of an icon in the icon database, as summarized in section Edit Menu on page SSR-5-8.
- View—contains operations that allow you to change the representation of the icon on your screen, as summarized in section View Menu on page SSR-5-9.
- Options—contains operations that allow you to change the font or colors used and to create or delete a transparency for the icon.

The Tool Palette

The Tool Palette displays when you right-click on the icon you want to edit and choose Edit from the pop-up menu.

Figure 5-1 Icon Editor Tool Palette



The Tool Palette contains the following editing tools for creating and editing an icon.

Table 5-3 Tool Palette Summary (Part 1 of 2)

Tool	Description
Hand	Repositions an oversize image. Select and drag the oversize image with the Hand Tool to reposition the image.
Dropper	Picks up the color from a pixel. Click anywhere in the image editing pane or the Color Palette to pick up a color. The color you pick up is displayed in the Foreground Color Box. To pick up a background color, press the Control key when you click the Dropper Tool.
Selection	Selects a rectangular area of an image for editing. To select the entire image editing pane, double-click the Selection Tool in the Tool Palette.
Lasso	Selects an irregular area.
Fill	Fills the enclosed area of an image with the foreground color.
Spray Can	Draws the foreground color with a “spray” pattern
Eraser	Replaces the color of an image with the background color. Drag the Eraser over the area you want to erase. Press the Shift key before erasing to constrain the Eraser Tool horizontally or vertically. To clear the entire editing area, click in the image editing pane, and then double-click the Eraser Tool in the Tool Palette.
Text	Places text into the image. Click in the image editing pane with the Text Tool to open an Image Text Window. Text entered in the Image Text Window appears as a selection in the image.
Pencil	Draws or erases one pixel at a time. Click in the image editing pane with the Pencil Tool to draw or erase a pixel. If the Pencil Tool is used on an image area that contains the foreground color, the Pencil Tool draws the background color. Otherwise, the Pencil Tool draws the foreground color. Drag the Pencil Tool to draw a continuous line. Use Fat Bits mode to draw one pixel at a time.
Line	Draws straight lines. Drag the Line Tool in the image editing pane to draw a straight line. Press the Shift key during the drag operation to constrain the line horizontally, vertically, or to a 45-degree angle.

Table 5-3 Tool Palette Summary (Part 2 of 2)

Tool	Description
Ellipse	Draws an ellipse. Drag the Ellipse Tool in the image editing pane to draw an ellipse. Press the Shift key during the drag operation to constrain the ellipse to a circle.
Filled Ellipse	Draws a ellipse filled with the foreground color. Drag the Filled Ellipse Tool in the image editing pane to draw a filled ellipse. Press the Shift key during the drag operation to constrain the ellipse to a circle.
Rectangle	Draws a rectangle. Drag the Rectangle Tool in the image editing pane to draw a rectangle. Press the Shift key during the drag operation to constrain the rectangle to a square.
Filled Rectangle	Draws a rectangle filled with the foreground color. Drag the Filled Rectangle Tool in the image editing pane to draw a filled rectangle. Press the Shift key during the drag operation to constrain the rectangle to a square.
Line width palette	Specifies the width of any lines drawn with the line tool or the border of an ellipse or rectangle.
Recently used colors	Contains the 16 colors most recently used in your image. When you select a new color, it is added to the bottom of the palette and a color is deleted from the top. To make a color in the palette the foreground color, left-click on it. To make it the background color, right-click on it.
Foreground/background colors	To select a new foreground or background color, click and hold the color box. When the Color Palette displays, drag the pointer to a new color.
End of Table 5-3	

File Menu

The File menu contains operations that relate to high-level functions.

Table 5-4 File Menu Summary

Menu item	Description	Reference
Close	Closes the Image Editor and places a copy of the edited icon in memory. When the icon database is saved (written to disk), the edited icon will be saved.	—
Import	Imports an image.	Importing an Image on page SSR-5-6
Export	Exports the image.	Exporting an Image on page SSR-5-7
Set Size	Sets the size of the image, in pixels. The size of the image editing pane does not change automatically when you change the size of the image. Use the zoom options to change the pane size so you will see the entire image in the pane.	—
Revert	Returns the image to the one shown when the Image Editor was opened (the version in the Icon Editor's memory).	—
End of Table 5-4		

Importing an Image

To import an image, follow these steps.

Procedure 5-1 Importing an Image

1. Choose File > Import…

   ➥ The Open dialog box displays.

2. Use the Filters option in the file chooser to specify an image format.

3. Select the file you want to import.

   ➥ The imported image is displayed in the Image Editor.

End of Procedure 5-1

Exporting an Image

To export an image, follow these steps.

Procedure 5-2 Exporting an Image

1. Choose File > Export…

   ➥ The Save As dialog box displays.

2. Specify the exported image format by selecting it from the Filters menu item.

3. Enter the exported file's name. You may want to include the format as part of the file name.

   ➥ The file is saved with a suffix that indicates the image format.

End of Procedure 5-2

Edit Menu

The Edit menu contains operations that allow you to change the appearance of an icon.

Table 5-5 Edit Menu Summary

Menu item	Description	Reference
Undo	Undoes the last editing operation.	—
Redo	Redoes the last undo.	—
Cut	Cuts the selected pixels, placing them in the clipboard.	—
Copy	Copies the selected pixels to the clipboard.	—
Paste	Pastes the pixels stored in the clipboard.	—
Clear	Deletes the selected pixels without storing them in the clipboard.	—
Rotate	Rotates the selected pixels one-quarter turn counter-clockwise.	—
Flip Horizontal	Flips the selected pixels around a vertical axis.	—
Flip Vertical	Flips the selected pixels around a horizontal axis.	—
Mask Selection	Masks the selected area, as described in the Create Transparency discussion. See Creating a Transparency on page SSR-5-13.	—
Crop to Selection	Crops to the selected area. Pixels outside the selection are discarded.	—
End of Table 5-5		

View Menu

The View menu contains operations that allow you to change the way the icon is displayed in the Image Editor.

Table 5-6 View Menu Summary (Part 1 of 2)

Menu item	Description	Reference
Fat Bits	Magnifies the image so each pixel can be viewed. When Fat Bits is disabled, the image returns to its actual size. (The default magnification is five times the size of the image.)	—
Draw From Center	Defines the starting point for certain drawing tools. Normally, the line, ellipse, and rectangle tools begin drawing an item from the end (or perimeter) of the item. Draw From Center specifies that drawing begins at the center of the item.	—
Show Grid	Displays the image in a grid. Individual pixels are outlined with the grid pattern. When the bit size displayed is smaller than three pixels, the grid is not shown.	—
Show Position	Displays the coordinates of an image. The origin of the image is the lower left corner. X: The position of the item on the X axis Y: The position of the item on the Y axis DX: The change in X as you draw or move an item DY: The change in Y as you draw or move an item Len: The distance between the starting point of the operation and the current position Ang: The arc, in degrees, described from the starting point as you draw or move an item	—

Table 5-6 View Menu Summary (Part 2 of 2)

Menu item	Description	Reference
Preview	Displays the entire image in a separate Preview Window. If you use the Hand Tool to pan through an oversize image, the visible area of the image is outlined in the Preview Window.	—
Zoom In	Magnifies the image. Select repeatedly to continually increase the magnification.	—
Zoom Out	Decreases the magnification of the image. Select repeatedly to continually decrease the magnification.	—
End of Table 5-6		

Options Menu

The Options menu contains operations that allow you to change the font or colors used and to create or delete a transparency for the icon.

Table 5-7 Options Menu Summary

Menu item	Description	Reference
Choose Font	Allows you to choose the font desired	—
Edit colors	Allows you to select, edit, and install a pre-defined palette, or to create and save a custom palette.	Color Palette Editor on page SSR-5-11
Create Transparency	Creates a pane where you define the transparency to be used with the icon.	Transparencies and Masks on page SSR-5-13
Delete Transparency	Deletes an existing transparency.	—
End of Table 5-7		

Color Palette Editor

Use the Color Palette Editor to

- Edit individual colors in the Color Palette
- Save edited color palettes
- Install different color palettes

Procedure 5-3 describes how to select and edit colors in a palette.

Procedure 5-3 Editing Colors in a Palette

1. Select the color or series of colors by clicking on the color in the palette or dragging the cursor over a series of colors.

   ➥ An outline appears around the selected colors.

2. Click on an editing button to start or complete an operation using the color(s) you selected (see Table 5-8 on page SSR-5-12).

End of Procedure 5-3

Table 5-8 lists the operations you can do with the editing buttons in the Color Palette.

Table 5-8 Editing Buttons

Button	Description
New	Creates a new color. You can then use the other buttons to edit the color.
Delete	Deletes the selected color.
Pick...	Brings up the Color Chooser. See the “Color Chooser” section in the User Interface chapter of the SP Sentinel User Guide.
Ramp	Creates a spectrum of shades (very light to very dark) of one color. This can be useful if you want to give subtle shading to a figure. See Creating and Installing a Shade Spectrum on page SSR-5-12 for instructions.
Blend	Creates a color that is a blend of two others. To create the blended color, select three contiguous colors, then click on the Blend button. The middle color becomes a blend of the two flanking colors.
Contrast	Changes the selected color to be its opposite on the color wheel. Red, for example, becomes green.
Lighter	Adds white to the selected color.
Darker	Adds black to the selected color.
Warmer	Adds a warm shade (red, orange, or yellow) to the selected color.
Cooler	Adds a cool shade (blue, purple, green) to the selected color.
Remap To Closest Color	Controls how the colors in an image are mapped when a new color palette is installed. If Remap To Closest Color is checked, the colors of the image are mapped to the closest colors in the newly-installed palette. This default behavior minimizes changes to an image when installing a new palette. If Remap to Closest Color is not checked, colors are mapped according to their position on the palette. For example, if the color in the top-left cell of the old palette is black, and the color in the same position in the new palette is green, all black pixels in the image become green.
End of Table 5-8	

Procedure 5-4 describes how to create a shade spectrum and install it as color palette.

Procedure 5-4 Creating and Installing a Shade Spectrum

1 Determine the number of different shades you want use (five in this example) and the color of the spectrum (orange in this example).

2 Dragging from left to right, select *seven* color cells, making sure that the left-most cell is orange.

3 Click on the Ramp button.

➥ The seven cells are recolored. The left-most cell is white. The right-most cell is black. The five cells in between range from light orange to dark orange.

4 Click on the Install button.

➥ The color palette is installed. The Color Palette Editor closes and the image in the Image Editor changes to reflect the newly installed color palette.

End of Procedure 5-4

Transparencies and Masks

To understand the role of transparencies, consider the following:

- An icon displays against the background of the object palette or editor.
- The icon itself is composed of the figure you designed, overlaid on a black rectangle.

The transparency is a black and white copy of your icon that sits in a separate layer underneath the icon. It controls what parts of the background are visible and which are hidden. White areas of the transparency allow the background to display. Black areas prevent the background from displaying, making your icon the visible layer.

One common use of a transparency is to mask the icon's black rectangle, as shown below.

Figure 5-2 Using a Transparency To Hide the Black Background

No transparency. Black rectangle of icon displays.

With transparency. Black rectangle is hidden, gray background displays.

The Mask Selection operation on the Edit menu helps you create an effective transparency.

Procedure 5-5 Creating a Transparency

1 Open the icon in the Image Editor.

➥ The image displays in one pane of the Image Editor.

2 Choose Options > Create Transparency.

➥ A second pane (the transparency pane) appears to the right of the first pane.

3 Use the selection tool to demarcate the area you want to mask.

4 Choose Edit > Mask Selection.

➥ A mask of the selection appears in the transparency pane.

5 Choose File > Close.

➥ The Image Editor closes and the newly-masked icon appears in the Icon Database Editor.

6 Choose File > Save to save the changed icon.

End of Procedure 5-5

6 Distributions

Distributions are used to generate a series of stochastic numeric values. A list of the predefined distributions appears in Table 6-1, along with their supported arguments. For more information on the mathematical framework of these distributions, consult the textbook *Fundamentals of Applied Probability Theory*, by Dr. Alvin Drake, published in 1967 by McGraw-Hill.

Table 6-1 Predefined Distributions (Part 1 of 2)

Name	First Argument	Second Argument
bernoulli	mean outcome	<none>
binomial	num_samples	p_success
chi_square	mean outcome	<none>
constant	outcome	<none>
erlang	scale	shape
exponential	mean outcome	<none>
extreme	location	mode_scale
fast_normal	mean outcome	outcome variance
gamma	scale	shape
geometric	prob_success	<none>
laplace	mean	scale
logistic	mean	scale
lognormal	mean outcome	outcome variance
normal	mean outcome	outcome variance
pareto	location	shape
poisson	mean outcome	<none>
power function	shape	scale
rayleigh	scale	<none>
triangular	min	max

Table 6-1 Predefined Distributions (Part 2 of 2)

Name	First Argument	Second Argument
uniform	minimum outcome (inclusive)	maximum outcome (exclusive)
uniform_int	minimum outcome (inclusive)	maximum outcome (inclusive)
Weibull	shape parameter	scale parameter
End of Table 6-1		

Figure 6-1 Bernoulli PMF Definition

$$P_x(x_0) = \begin{cases} (1 - P) & x_0 = 0 \\ P & x_0 = 1 \\ 0 & otherwise \end{cases}$$

$$0 < P < 1$$

mean: $E(x) = P$

variance: $\sigma_x^2 = P(1 - P)$

Arg 1: mean_outcome = P
Arg 2: *<none>*

Figure 6-2 Bernoulli PMF Shape

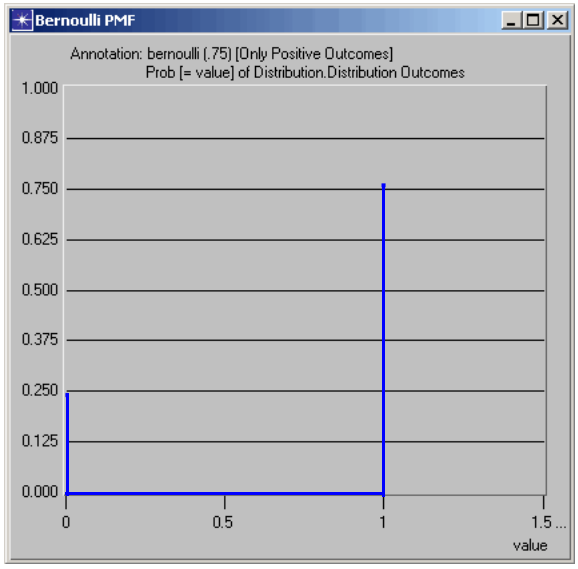


Figure 6-3 Binomial PMF Definition

$$f(x) = \binom{n}{x} p^x q^{n-x} \qquad 0 \le x \le n \quad \textit{(x an integer)}$$

$$\text{mean: } E(x) = np$$

$$\text{variance: } \sigma_x^2 = npq$$

$$q = 1 - p$$

Arg 1: num_samples = n (positive integer)
Arg 2: p_success = p where 0 < p < 1

Figure 6-4 Binomial PMF Shape

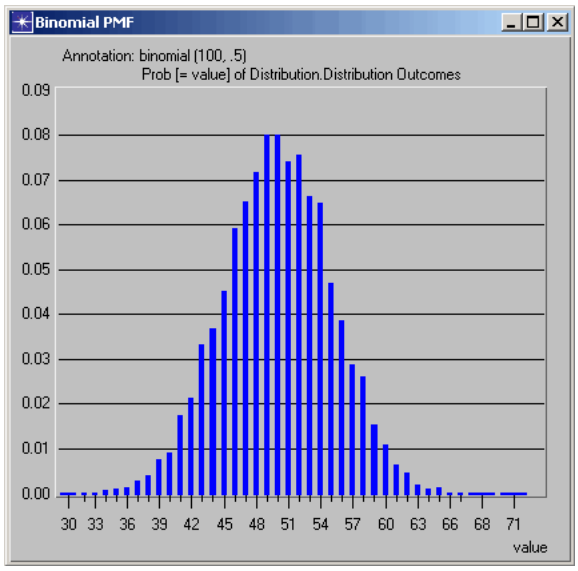


Figure 6-5 Chi-Square PDF Definition

$$f_x(x_0) = \begin{cases} \left[\left(\frac{n}{2}-1\right)!\right]^{-1} 2^{-n/2} x_0^{(n/2)-1} e^{-x_0/2} & x_0 > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$n = 1, 2, 3, \ldots$$

mean: $E(x) = n$

variance: $\sigma_x^2 = 2n$

Arg 1: mean_outcome = n
Arg 2: <none>

Figure 6-6 Chi-Square PDF Shape

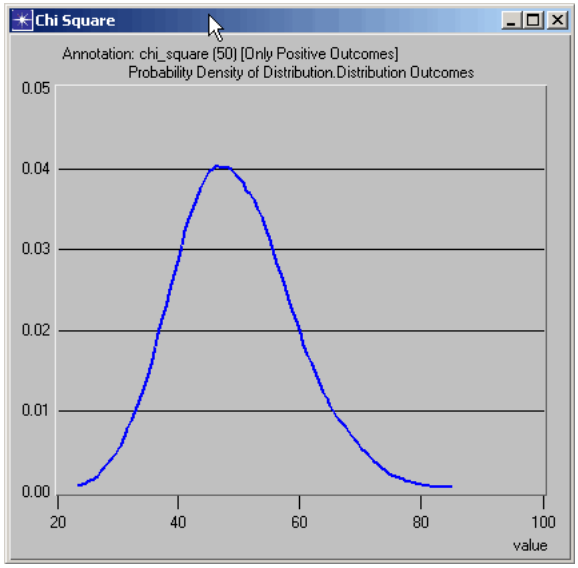


Figure 6-7 Constant PMF Definition

$$P_x(x_0) = \begin{cases} 1 & x0 = C \\ 0 & otherwise \end{cases}$$

mean: $E(x) = C$

variance: $\sigma_x^2 = 0$

Arg 1: outcome = C
Arg 2: <none>

Figure 6-8 Constant PMF Shape

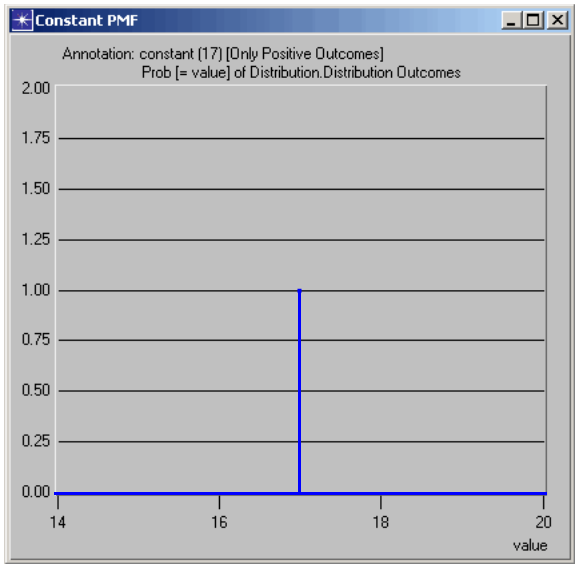


Figure 6-9 Erlang PDF Definition

$$f_x(x_0) = \begin{cases} \left(\frac{x_0}{b}\right)^{c-1} e^{-\frac{x_0}{b}} / (b(c-1)!) & x_0 > 0 \\ 0 & \text{otherwise} \end{cases}$$

$b > 0 \qquad c = 1, 2, 3, \ldots$

mean: $E(x) = bc$

variance: $\sigma_x^2 = c(b^2)$

Arg 1: scale = b
Arg 2: shape = c

Figure 6-10 Erlang PDF Shape

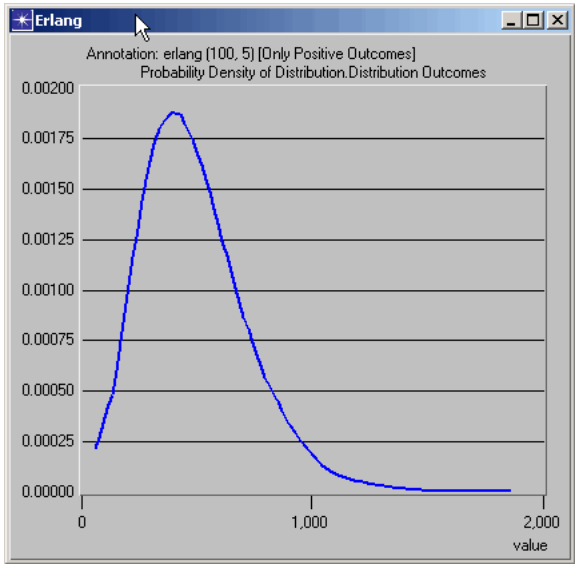


Figure 6-11 Exponential PDF Definition

$$f_x(x_0) = \begin{cases} ae^{-ax_0} & x_0 > 0 \\ 0 & \text{otherwise} \end{cases}$$

$a > 0$

mean: $E(x) = a^{-1}$

variance: $\sigma_x^2 = a^{-2}$

Arg 1: mean_outcome = 1 / a
Arg 2: <none>

Figure 6-12 Exponential PDF Shape

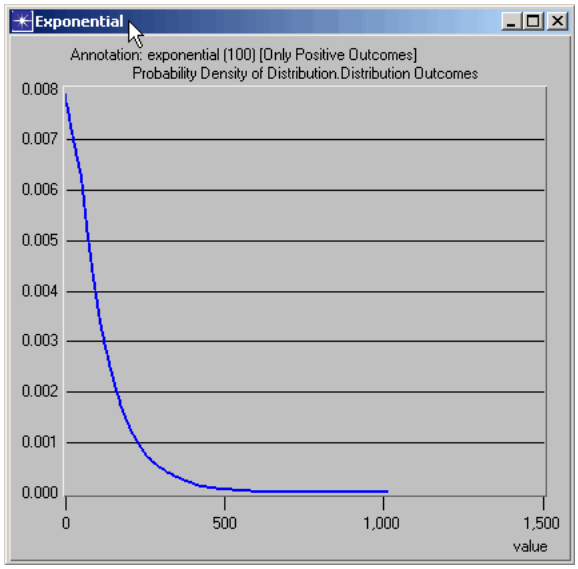


Figure 6-13 Extreme Value (Gumbel) PDF Definition

$$f(x) = (1/b)\exp[-(x-a)/b] \times \exp\{-\exp[-(x-a)/b]\} \qquad -\infty < x < \infty$$

Arg 1: location = a
Arg 2: scale = b > 0

mean: $E(x) = a - b\Gamma'(1)$

variance: $\sigma_x^2 = b^2\Pi^2/6$

$\Gamma'(1) = -0.57721$ $\Gamma(1)$ is the first derivative of the gamma function with respect to n at n = 1

Figure 6-14 Extreme Value (Gumbel) PDF Shape

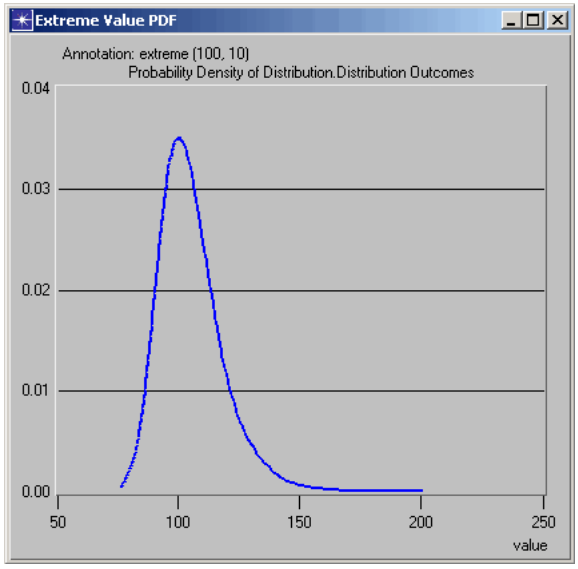


Figure 6-15 Gamma PDF Definition

$$f(x) = (x/b)^{c-1}[\exp(-x/b)]/b\Gamma(c) \qquad 0 \le x < \infty$$

$\Gamma(c)$ *is the gamma function with argument c*

mean: $E(x) = bc$

variance: $\sigma_x^2 = b^2c$

Arg 1: scale = b > 0
Arg 2: shape = c > 0

Figure 6-16 Gamma PDF Shape

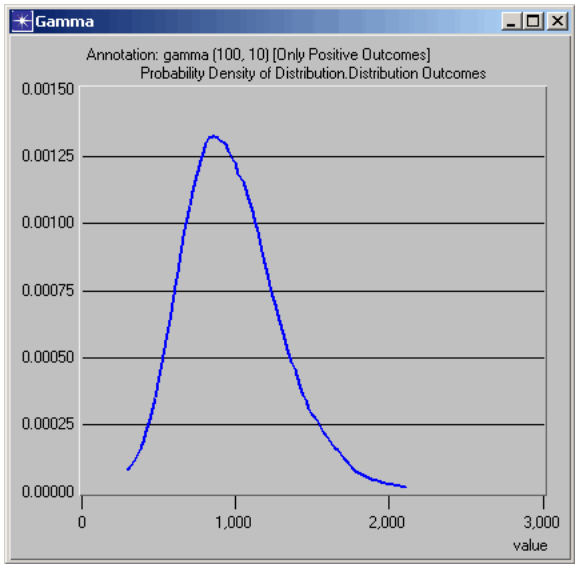


Figure 6-17 Geometric PMF Definition

$$f(x) = pq^x \qquad x \geq 0 \quad \textit{(x an integer)}$$

mean: $E(x) = q/p$

variance: $\sigma_x^2 = q/p^2$

$$q = 1 - p$$

Arg 1: prob_success = 0 < p < 1
Arg 2: (ignored)

Figure 6-18 Geometric PMF Shape

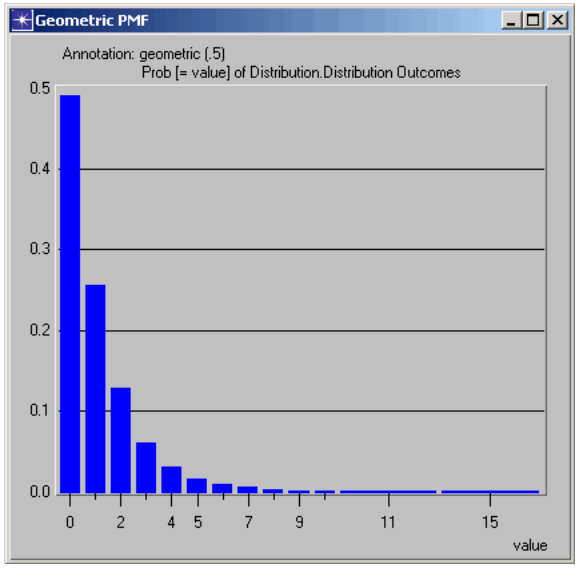


Figure 6-19 Laplace (Double Exponential) PDF Definition

$$f(x) = \frac{1}{2b}\exp\left(-\frac{|x-a|}{b}\right) \qquad -\infty < x < \infty$$

mean: $E(x) = a$

variance: $\sigma_x^2 = 2b^2$

Arg 1: mean = a (all real)
Arg 2: scale = b > 0

Figure 6-20 Laplace (Double Exponential) PDF Shape

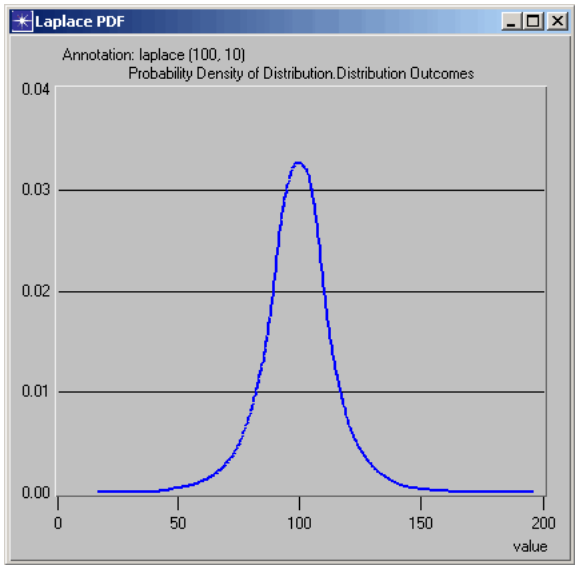


Figure 6-21 Logistic PDF Definition

$$f(x) = \frac{\exp[(x-a)/b]}{b\{1+\exp[(x-a)/b]\}^2} \qquad -\infty < x < \infty$$

mean: $E(x) = a$

variance: $\sigma_x^2 = \pi^2 b^2/3$

Arg 1: mean = a
Arg 2: scale= b > 0

Figure 6-22 Logistic PDF Shape

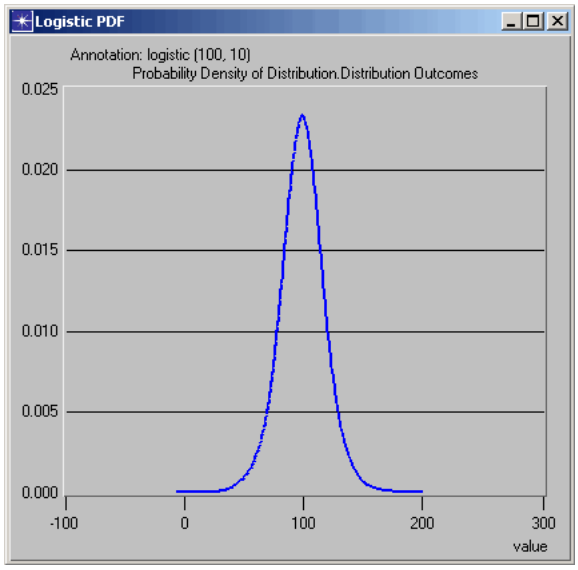


Figure 6-23 Lognormal PDF Definition

$$f(x) = \begin{cases} \dfrac{1}{x\sqrt{2\pi\sigma^2}} \exp \dfrac{-(\ln x - \mu)^2}{2\sigma^2} & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Arg 1: mean = $e^{\mu + \sigma^2/2}$

Arg 2: outcome_variance = $(e^{2\mu + \sigma^2})(e^{\sigma^2} - 1)$

Figure 6-24 Lognormal PDF Shape

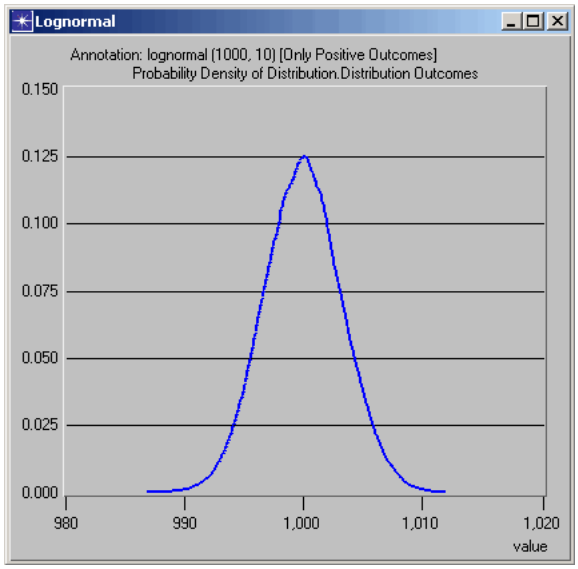


Figure 6-25 Normal and Fast Normal PDF Definition

$$f_x(x_0) = \frac{e^{-([x_0 - E(x)]^2/(2\sigma_x^2))}}{\sqrt{2\pi}\sigma_x} \qquad all\ x_0$$

$$-\infty < E(x) < \infty$$

$$\sigma_x > 0$$

Arg 1: mean_outcome = E(x)

Arg 2: outcome_variance = σ^2

Note—fast_normal and normal model the same distribution. The normal distribution is calculated using the Box-Muller Transform method. It provides greater resolution than fast_normal, but may result in slower simulations. The fast_normal distribution is table-based, with results truncated beyond 10 standard deviations. It is provided for speed and backward compatibility with releases prior to 6.0, in which it was the normal distribution.

Figure 6-26 Normal and Fast Normal PDF Shape

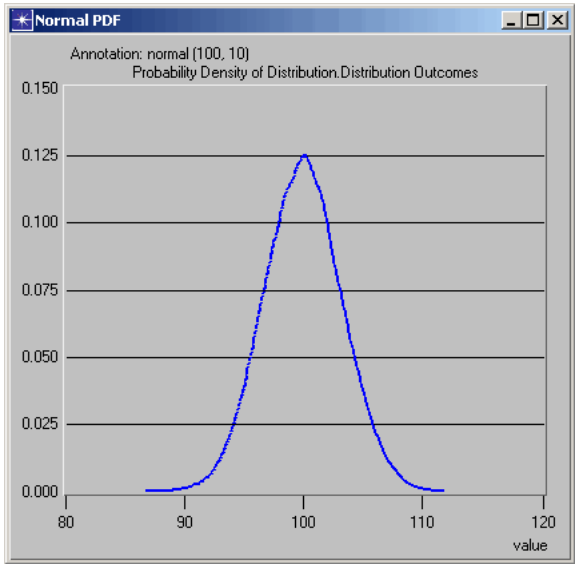


Figure 6-27 Pareto PDF Definition

$$f(x) = ca^c / x^{c+1} \qquad a \le x < \infty$$

mean: $E(x) = ca/(c-1)$ $c > 1$

variance: $\sigma_x^2 = ca^2/[(c-1)^2(c-2)]$ $c > 2$

Arg 1: location = a > 0
Arg 2: shape = c > 0

Figure 6-28 Pareto PDF Shape

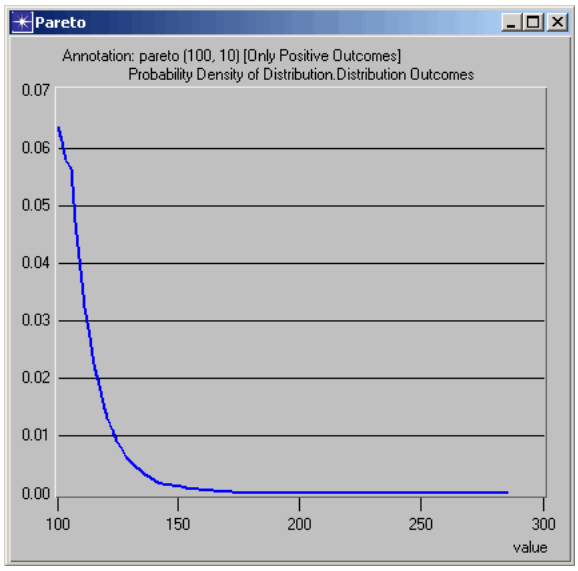


Figure 6-29 Poisson PMF Definition

$$p_x(x_0) = \frac{a^{x_0} e^{-a}}{x_0!} \qquad x_0 = 0, 1, 2, \ldots$$

$a > 0$

mean: $E(x) = a$

variance: $\sigma_x^2 = a$

Arg 1: mean_outcome = a
Arg 2: <none>

Figure 6-30 Poisson PMF Shape

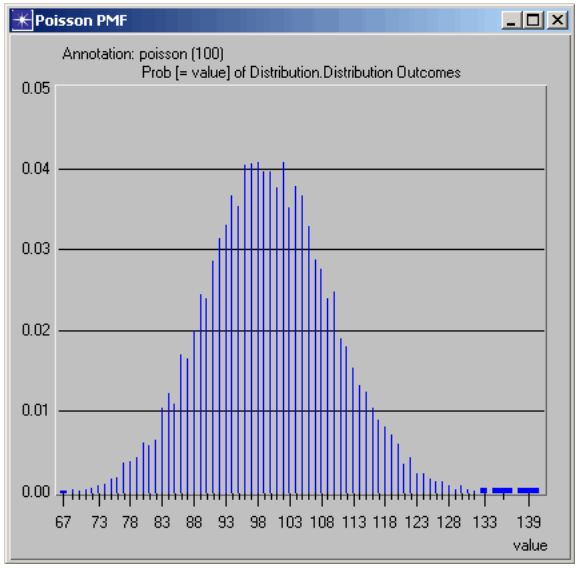


Figure 6-31 Power Function PDF Definition

$$f(x) = \frac{cx^{c-1}}{b^c} \qquad 0 \le x \le b$$

mean: $E(x) = bc/(c+1)$

variance: $\sigma_x^2 = b^2c/[(c+2)(c+1)^2]$

Arg 1: shape = c
Arg 2: scale = b > 0

Figure 6-32 Power Function PDF Shape

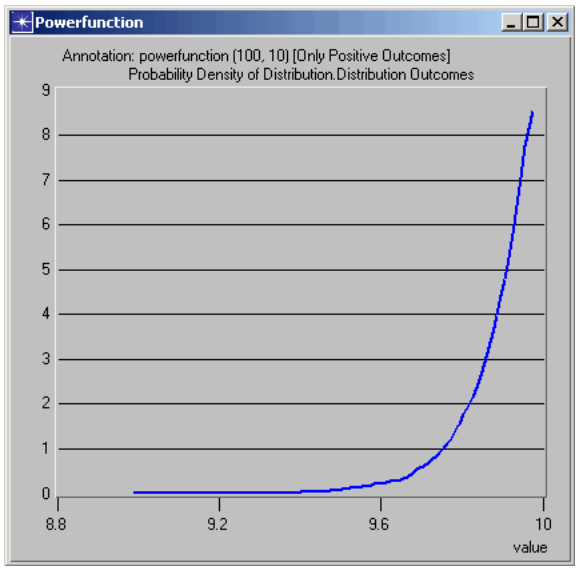


Figure 6-33 Rayleigh PDF Definition

$$f(x) = (x/b^2)\exp[-x^2/(2b^2)] \qquad 0 < x < \infty$$

mean: $E(x) = b(\pi/2)^{\frac{1}{2}}$

variance: $\sigma_x^2 = (2-\pi/2)b^2$

Arg 1: scale = b > 0
Arg 2: *<none>*

Figure 6-34 Rayleigh PDF Shape

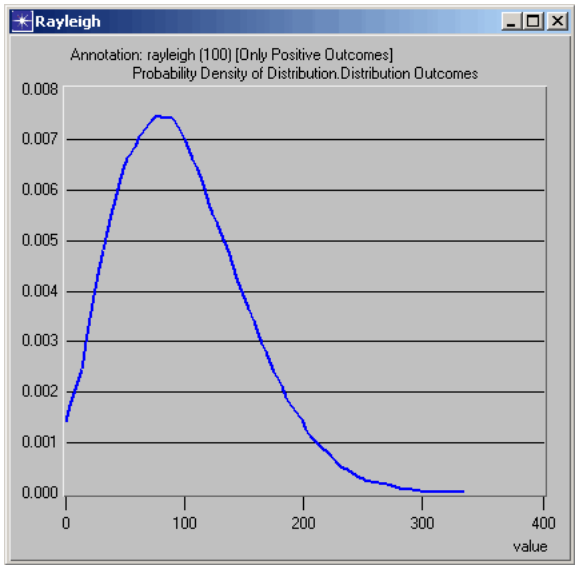


Figure 6-35 Triangular (Symmetrical) PDF Definition

$$f(x) = \begin{cases} \dfrac{4(x-a)}{(b-a)^2} & a \le x \le \dfrac{a+b}{2} \\ \dfrac{4(b-x)}{(b-a)^2} & \dfrac{a+b}{2} \le x \le b \end{cases}$$

mean: $E(x) = (a+b)/2$

variance: $\sigma_x^2 = \dfrac{(a-b)^2}{24}$

Arg 1: location lower limit =a
Arg 2: location upper limit = b

Figure 6-36 Triangular (Symmetrical) PDF Shape

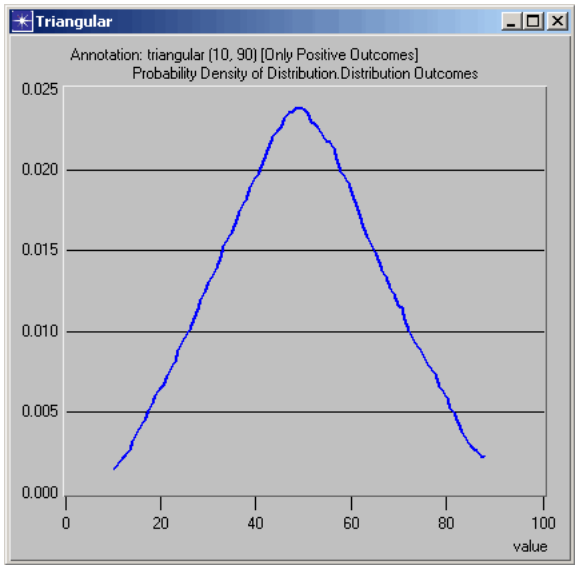


Figure 6-37 Uniform PDF Definition

$$f_x(x_0) = \begin{cases} 1/(b-a) & a \le x_0 < b \\ 0 & \text{otherwise} \end{cases}$$

$$-\infty < a < b < \infty$$

mean: $E(x) = (a + b)/2$

variance: $\sigma_x^2 = (b-a)^2/12$

Arg 1: min_outcome = a
Arg 2: max_outcome = b

Figure 6-38 Uniform PDF Shape

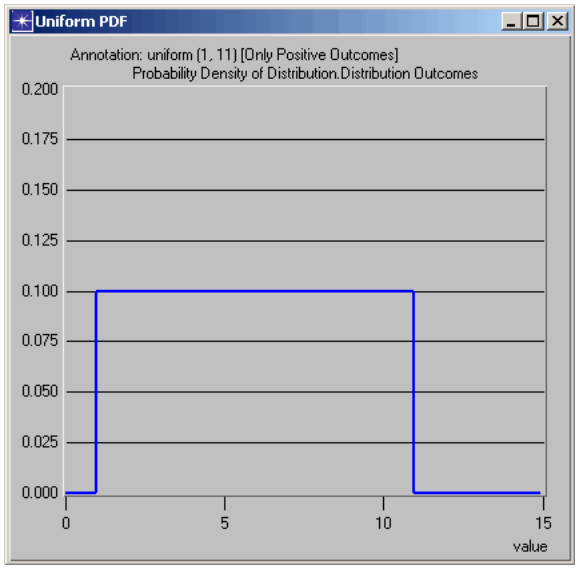


Figure 6-39 Uniform Integer PMF Definition

$$p_x(x_0) = 1/(1 + b - a) \qquad x_0 = 0, \pm 1, \pm 2, \ldots$$

$$a \le x_0 \le b$$

$a = 0, \pm 1, \pm 2, \ldots$

$b = 0, \pm 1, \pm 2, \ldots \qquad a \le b$

mean: $E(x) = (a + b)/2$

variance: $\sigma_x^2 = \frac{1}{12}[(b - a)(2 + b - a)]$

Arg 1: min_outcome = a
Arg 2: max_outcome = b

Figure 6-40 Uniform Integer PMF Shape

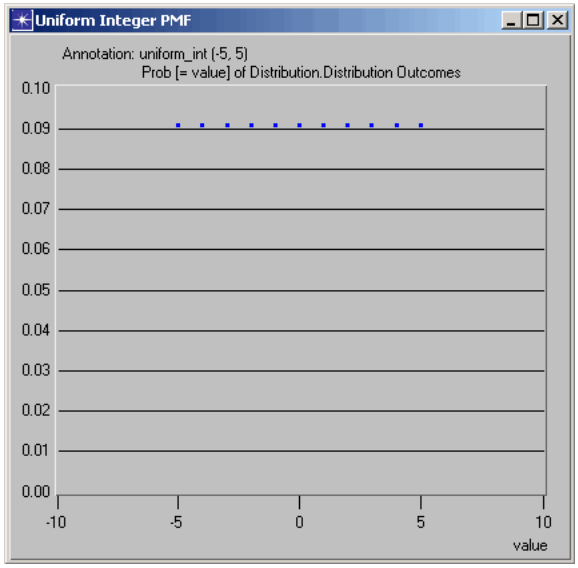


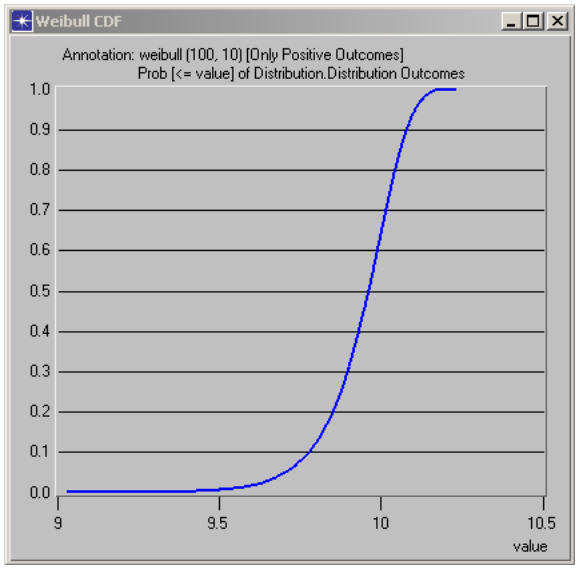
Figure 6-41 Weibull CDF Definition

$$f(x) = \begin{cases} 1 - e^{-\left(\frac{x}{\beta}\right)^{\alpha}} & \text{if}(x > 0) \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha > 0$
and $\beta > 0$

Arg 1: shape = α
Arg 2: scale = β

Figure 6-42 Weibull CDF Shape



Index

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X