Contents

Preface
Who Should Use This Guide................................................................. 8
Summary of Contents........................................................................... 9
What Typographic Variations Mean ..................................................... 10

Chapter 1
The OPSEC Environment
Overview............................................................................................ 14
VPN-1 Architecture and the OPSEC Model.......................................... 15
Getting Started.................................................................................... 18
OPSEC API Overview.......................................................................... 19
  Environment Variables...................................................................... 22
  Debugging....................................................................................... 24
  OPSEC Errors................................................................................ 24
  OPSEC Data Structures and Functions.............................................. 24
  Session Multiplexing..................................................................... 32
  OPSEC Unique ID.......................................................................... 32
  OPSEC Values............................................................................... 32
  Multithreaded OPSEC Applications................................................. 33
Client-Server Connection.................................................................... 37
  Establishing an authentication key............................................... 40
  Pulling Certificates...................................................................... 41
  Enabling SIC............................................................................... 43
Configuring the OPSEC Application.................................................... 44
Multi-SIC.......................................................................................... 55
Multi-SIC Configuration................................................................. 55
Platforms Compatibility..................................................................... 58

Chapter 2
OPSEC API (Application Programming Interface)
General OPSEC Functions.................................................................. 60
  Managing Environments.................................................................. 60
  Managing Entities.......................................................................... 69
  Managing Sessions........................................................................ 75
  Passing Information Between Entities........................................... 81
  Managing the Main Loop, Scheduling, and Socket Events............... 84
OPSEC Utilities................................................................................ 89
SIC Utilities..................................................................................... 99
Macros............................................................................................ 105
Error Handling Functions............................................................... 106
Events API Functions...................................................................... 108
  Managing Events........................................................................ 108
  Raising and Unraising Event Instances.................................... 112
OPSEC Unique IDs......................................................................... 116
OPSEC Value API.......................................................................... 120
Preface

In This Chapter

Who Should Use This Guide .............................................................. page 8
Summary of Contents ................................................................ page 9
What Typographic Variations Mean ............................................... page 10
Who Should Use This Guide

This document describes the OPSEC API Specification.

This API specification is written for developers who write software to enhance the network security provided by VPN-1.

It assumes that you have read the Check Point VPN-1 OPSEC API Specification.

It also assumes that you have a basic understanding and a working knowledge of the following:

- system and network security
- the VPN-1 product
- system and network administration
- the C and/or C++ programming language
- the Unix or Windows operating system
- Internet protocols
Summary of Contents

This guide contains the following chapters:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Description</th>
</tr>
</thead>
</table>
| Chapter 1, “The OPSEC Environment” | A technically oriented introduction to OPSEC, that includes:  
  • a brief overview of the OPSEC programming model  
  • an overview of how the OPSEC API is used by a Server or Client application  
  • a description of how to develop an OPSEC application |
| Chapter 2, “OPSEC API (Application Programming Interface)” | Describes each of the common OPSEC API functions in detail. The function prototypes are defined in the opsec.h file. |
What Typographic Variations Mean

The following table describes the typographic variations used in this book.

<table>
<thead>
<tr>
<th>Typeface or Symbol</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>AaBbCc123</td>
<td>The names of commands, files, and directories; on-screen computer output; code</td>
<td>Edit your .login file. Use <code>ls -a</code> to list all files. <code>machine_name%</code> You have mail. <code>session = sam_new_session (client, server);</code></td>
</tr>
<tr>
<td>AaBbCc123</td>
<td>same as above, but with emphasis</td>
<td><code>session = sam_new_session (client, server);</code></td>
</tr>
<tr>
<td>Save</td>
<td>Text that appears on an object in a window</td>
<td>Click on the <code>Save</code> button.</td>
</tr>
<tr>
<td>&lt;your text&gt;</td>
<td>Replace the angle brackets and the text they contain with your text.</td>
<td>Edit the file <code>&lt;FWDIR&gt;\lib\yourfile.xxx</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>line 1</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>line 2</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>.*</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>line n</code></td>
</tr>
</tbody>
</table>
### TABLE P-1  Typographic Conventions (continued)

<table>
<thead>
<tr>
<th>Typeface or Symbol</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>[item]</td>
<td>The item is optional.</td>
<td>dir [/o]</td>
</tr>
<tr>
<td>[item1] ... [item2]</td>
<td>List of optional items</td>
<td>dir [/o] [/w] [/s]</td>
</tr>
<tr>
<td>item1</td>
<td>item2</td>
<td>item3</td>
</tr>
<tr>
<td>italic</td>
<td>Specific values will be shown in italics</td>
<td>one of addnet</td>
</tr>
</tbody>
</table>
# Chapter 1

## The OPSEC Environment

In This Chapter

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>14</td>
</tr>
<tr>
<td>VPN-1 Architecture and the OPSEC Model</td>
<td>15</td>
</tr>
<tr>
<td>Getting Started</td>
<td>18</td>
</tr>
<tr>
<td>OPSEC API Overview</td>
<td>19</td>
</tr>
<tr>
<td>Client-Server Connection</td>
<td>37</td>
</tr>
</tbody>
</table>
Check Point's OPSEC (Open Platform for Security) integrates and manages all aspects of network security through an open, extensible management framework. Third party security applications can plug into the OPSEC framework via published application programming interfaces (APIs). Once integrated into the OPSEC framework, all applications can be configured and managed from a central point, utilizing a single Security Policy editor.

The OPSEC framework allows users to quickly exploit the latest developments in security technology and to choose system components from both Check Point and other security vendors that best meet their requirements. With OPSEC, all facets of network security are defined and driven by a single, central enterprise-wide Security Policy, ensuring that corporate security is safer, simpler to maintain and comprehensive.

This document is a technically oriented introduction to OPSEC, and includes:

- a brief overview of the OPSEC programming model
- an overview of how the OPSEC API is used by a Server or Client application
- a description of how to develop an OPSEC application
VPN-1 Architecture and the OPSEC Model

VPN-1 is comprised of two primary modules:

- **VPN-1 SmartCenter Server**
  
  SmartCenter Server manages the VPN-1 database, including the Rule Base, network objects, services, users etc.

- **VPN-1 Module**
  
  VPN-1 Module includes the Inspection Module, VPN-1 daemons and Security Servers.
  
  VPN-1 Module implements the Security Policy, logs events, and communicates with SmartCenter Server using the daemons.

The system administrator defines an enterprise Security Policy with SmartCenter Server (using one of the available SMART Clients) and enforces that Security Policy at various enforcement points. Enforcement points are machines on which the VPN-1 Module is installed, such as gateways, routers, and switches. The VPN-1 Module allows or disallows communication attempts, encrypts and decrypts packets, performs address translation, load balancing, content security checking, etc.

*Figure 1-1* shows a distributed Client/Server configuration. In this configuration, SmartCenter Server controls three VPN-1 Modules, each on a different platform, which in turn protect three heterogeneous networks.
OPSEC integrates all aspects of network security within a single, extensible framework. This framework provides central configuration and management for VPN-1 while integrating third-party security applications. The enterprise security system is composed of several components, each of which may be provided by a different vendor and installed on a different machine. This allows you to choose the system components from both Check Point and other security vendors that best meet the needs of your enterprise.

The OPSEC framework's modularization gives you a high degree of flexibility and enables you to do all of the following:

- Exploit the latest developments in security technology.
- Upgrade one component without changing the others. For example, an anti-virus server may require frequent updates as new viruses are discovered.
- Resolve platform incompatibility problems.
Distribute the processing load by having the VPN-1 Module “outsource” some of its security tasks to other components.

**Note** - It may be necessary to configure each OPSEC component to make it aware of the other components. For example, a VPN-1 Module will communicate with another component only if there is a rule in its Rule Base that allows it to do so.

The Check Point OPSEC Software Development Kit (SDK) provides Application Programming Interfaces (APIs) for open protocols. The OPSEC APIs can be used to configure transactions between VPN-1 and OPSEC components. The OPSEC SDK includes the following APIs:

- **CVP (Content Vectoring Protocol)** used to implement content screening and anti-virus checking.
- **UFP (URL Filtering Protocol)** used to control access to external Web sites.
- **SAM (Suspicious Activity Monitoring)** used to detect and block intrusion attempts.
- **LEA (Log Export API)** used to retrieve and export VPN-1 Log data.
- **ELA (Event Logging API)** used to enable third-party applications to log events into the VPN-1 SmartCenter.
- **UserAuthority** used to provide network security information to third-party applications.
- **AMON (Application Monitoring API)** used to enable third-party applications to export their status information to VPN-1.
- **CPMI (Check Point Management Interface)** used to provide a secure interface to the Check Point VPN-1 SmartCenter Server and its components.
- **CSA (Cluster Status API)** used to enable third-party applications to monitor and affect the ClusterXL state and configuration.
- **SmartUpdate OPSEC Interface** to remotely manage (install/uninstall) the third-party application.
Before you begin developing OPSEC applications, it is recommended that you do the following:

1. Familiarize yourself with Check Point VPN-1 by reading the *Check Point VPN-1 Administration Guide*.
   As a developer, you must understand how your application is to be defined and used with VPN-1.

2. Read this document.

3. Read the API specification for the OPSEC task (CVP, UFP etc.) you are developing. For information on Provider-1 see the “*Check Point Provider-1 User Guide*”.
   These documents are also available on the Check Point OPSEC support page at http://www.opsec.com.

4. Acquire the necessary software libraries from Check Point.
   Information on how to acquire the OPSEC software libraries is available on the Check Point OPSEC support page at http://www.opsec.com.

5. Check out the most up-to-date information about OPSEC and the latest version of the Release Notes at http://www.opsec.com.
OPSEC API Overview

OPSEC provides a powerful message based, layered environment. The OPSEC API defines an asynchronous interface suitable for developing:

- servers that implement one or more OPSEC security tasks
- clients that use an OPSEC Server

Note - In the OPSEC environment, the distinction between Client and Server is that the Client locates and initiates the connection to the Server.

The OPSEC API includes the functions used for opening and monitoring connections between Client and Server. Communication between OPSEC Clients and Servers is implemented using Check Point's Secure Internal Communications (SIC) infrastructure. The OPSEC Transport Layer converts the messages sent across the connections into events. These functions are common to all OPSEC applications.

In addition to the common API functions used by all OPSEC applications, specific applications use specific API functions. This is illustrated in Figure 1-2.

![OPSEC API hierarchy](image)

**Figure 1-2** OPSEC API hierarchy

NOTE: The Client and Server processes can also be the same process.
Note that even specific OPSEC services are, in fact, quite general. For example, a CVP Server analyzes a data stream it receives from a CVP Client and returns information based on its analysis. This model is suitable for anti-virus checking, user authentication and statistical analyses of text, among other functions. The “meaning” of messages and events in the OPSEC model is determined by specific OPSEC Clients and Servers.

At the heart of an OPSEC application is an endless loop that waits for events to occur and then processes them. Events are handled by the OPSEC API functions, which may call user-defined functions to further process events.

Figure 1-3 illustrates this structure.

Figure 1-3  OPSEC Application Structure

```c
OMI_HANDLE ld;
...
ld = omi_new_session(..., after_open_callback(),...);
if (ld == NULL) {
  ...
}

int after_open_callback(..., rc)
{
  if (rc == ...) {
    ...
  }
  i = OPSEC_ldap_bind(..., after_bind_callback(),...);
  if (i == ...) {
    ...
  }
}

int after_bind_callback(..., rc)
{
  if (rc == ...) {
    ...
  }
}
```
The OPSEC environment is a framework in which OPSEC applications communicate. Each OPSEC process creates only one OPSEC environment.

An OPSEC entity is an instantiation of a specific behavior. In practice, an entity defines a behavior by specifying the handler functions for each type of event it receives. An OPSEC environment can simultaneously contain any number of OPSEC entities.

An OPSEC session is a dialog between two OPSEC entities. There are two types of OPSEC sessions:

- generic OPSEC sessions
  These sessions enable OPSEC entities to exchange service-independent messages. For example, an OPSEC entity may use a generic session to “ping” its peer and receive the round-trip response time.

- service-specific OPSEC sessions
  These sessions extend the functionality of generic OPSEC sessions to enable OPSEC entities to exchange service-specific messages (i.e. messages specific to LEA, CVP, etc.).

  The functions that create service-specific OPSEC sessions are different for each OPSEC service.

An OPSEC entity can simultaneously control any number of OPSEC sessions.

The relationship between these elements is illustrated in Figure 1-4.
Suppose that an OPSEC application on machine A is running both a LEA Client and a SAM Client. The LEA Client entity checks the logs of the processes running on machines B and C by establishing sessions with the LEA Server entities on those machines. If the LEA Client entity discovers something suspicious, the OPSEC application can call on the SAM Client entity. The SAM Client entity then establishes a session with the SAM Server entity running on machine C, which handles the suspicious activity.

**Environment Variables**

**Behavioral Changes**

The behavior of the OPSECDIR environment variable and its effect on the search path for OPSEC configuration files have changed in Next Generation. The following side effects may be expected:

- The key files for SSL (Secure Socket Layer) and FWN1 based communication will NOT be affected. The same behavior of the opsec_putkey will be observed.
- If the name specified is relative, the configuration file(s) (the name of which is supplied to the opsec_init() API via the OPSEC_CONF_FILE attribute) will now be loaded relative to OPSECDIR and not relative to the working directory.
If the specified name(s) is absolute, opsec_init() will load them from the path specified by the name with no regard to OPSECDIR or the working directory (this part of the behavior is the same as in previous versions).

**OPSECDIR**

The OPSECDIR environment variable can be used to specify the directory where the OPSEC application will maintain configuration files. This variable must specify the name of an existing directory.

There is a limit of 512 characters for the total length of the file name (including OPSECDIR). If OPSECDIR is not defined, the OPSEC application uses the working directory.

The location of the following configuration files is affected by the OPSECDIR environment variables:

- The SSL and FWN1 authentication key files
  These will be placed in OPSECDIR at the time of creation, by the opsec_putkey tool and will be searched for in OPSECDIR at the time of OPSEC environment creation.

The next three files will be searched for differently depending upon relative or absolute path and file names. If the specified path and file name is relative (meaning it does not start with a backslash "\", a drive letter and colon in Windows NT or slash "/" in UNIX) it will be searched for in relation to OPSECDIR. If the specified name is not relative (i.e. it is absolute), OPSECDIR will not be used.

- The configuration file specified at the OPSEC environment creation time
- The SSLCA certificate file
- The SIC Policy file

In addition, OPSECDIR will contain some other temporary files which are created by OPSEC at run-time, mainly for SIC.

**OPSEC_DEBUG_LEVEL**

See “Debugging” below for information about this environment variable.
Debugging

To receive OPSEC debugging information during runtime, set the environment variable `OPSEC_DEBUG_LEVEL` to a value between 0 (no debugging information) and 3 (all debugging information) before the application starts. Debugging information is output to stderr.

In addition to the normal OPSEC debug you may view CPMI debug information by setting the `OPSEC_DEBUG_LEVEL` environment variable to values between 4 (only fatal CPMI errors) and 9 (all CPMI information).

OPSEC Errors

There are two kinds of OPSEC errors:

- general errors
  For these errors, OPSEC sets the global `opsec_errno` variable. To determine the meaning of the error, use the `opsec_errno_str` function.

- SIC errors
  These errors may occur when using SIC to establish a connection. To determine the meaning of the error, use the `opsec_get_sic_error` function.

Note that SIC is not used when a connection is established using the `port` keyword. See “Client-Server Connection” on page 37 for details.

<table>
<thead>
<tr>
<th>function name</th>
<th>description</th>
<th>See ...</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>opsec_errno_str</code></td>
<td>converts the OPSEC error number to a string</td>
<td>page 10</td>
</tr>
<tr>
<td><code>opsec_get_sic_error</code></td>
<td>retrieves SIC errors that occurred when establishing a connection</td>
<td>page 10</td>
</tr>
</tbody>
</table>

OPSEC Data Structures and Functions

Information about OPSEC environments, entities and sessions is stored in data structures that are specific to each element. You can use the OPSEC API functions to manage and retrieve information from these structures. These data structures and functions are common to all OPSEC applications.
Environment Structure and Functions

Each OPSEC application stores information about its environment and entities in an OpsecEnv structure. This information includes configuration information (see “Configuring the OPSEC Application” on page 44).

The functions that handle the OPSEC Environment are listed in Table 1-2.

<table>
<thead>
<tr>
<th>function name</th>
<th>description</th>
<th>See ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>opsec_init</td>
<td>initializes an OpsecEnv structure</td>
<td>page 60</td>
</tr>
<tr>
<td>opsec_env_destroy</td>
<td>destroys an OpsecEnv structure</td>
<td>page 64</td>
</tr>
<tr>
<td>opsec_get_conf</td>
<td>retrieves a value from an OpsecEnv structure</td>
<td>page 67</td>
</tr>
</tbody>
</table>

Entity Structure and Functions

The OpsecEntity structure stores information about a given entity, such as the entity’s name and its type (e.g. CVP Server). You can use this structure to define communication parameters such as port number, type, etc.

The functions that handle the OpsecEntity structure are listed in Table 1-3.

<table>
<thead>
<tr>
<th>function name</th>
<th>description</th>
<th>See ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>opsec_init_entity</td>
<td>defines an entity in the OPSEC environment</td>
<td>page 69</td>
</tr>
<tr>
<td>opsec_destroy_entity</td>
<td>destroys an entity</td>
<td>page 74</td>
</tr>
</tbody>
</table>

Session Structure and Functions

Information about the current session is stored in the OpsecSession structure.

The names of the functions that create OPSEC service-specific sessions are of the form xxx_new_session, where xxx is an OPSEC service (for example, LEA, SAM etc.). These functions are described in detail in the corresponding documentation (e.g. the LEA (Log Export API) Specification).

An OPSEC Client initiates a service-specific session by calling one of the xxx_new_session functions. To initiate a generic session, the OPSEC Client calls opsec_new_generic_session.
An OPSEC Server enables Clients to start sessions with it by calling `opsec_start_server`.

Sessions are terminated by the underlying OPSEC level.

**Note** - After a session is terminated, the `OpsecSession` structure is no longer valid. Any attempt to access or use the structure will result in undefined behavior.

The functions that manage sessions and retrieve information from the `OpsecSession` structure are listed in Table 1-4.

**Table 1-4**  
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
<th>See ...</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>opsec_new_generic_session</code></td>
<td>starts a generic OPSEC session</td>
<td>page 75</td>
</tr>
<tr>
<td><code>opsec_start_server</code></td>
<td>starts the OPSEC Server</td>
<td>page 76</td>
</tr>
<tr>
<td><code>opsec_stop_server</code></td>
<td>suspends the OPSEC Server</td>
<td>page 76</td>
</tr>
<tr>
<td><code>opsec_suspend_session_read</code></td>
<td>stops the peer from sending events on all sessions</td>
<td>page 77</td>
</tr>
<tr>
<td><code>opsec_resume_session_read</code></td>
<td>allows the peer to resume sending events</td>
<td>page 77</td>
</tr>
<tr>
<td><code>opsec_end_session</code></td>
<td>terminates the OPSEC session</td>
<td>page 78</td>
</tr>
<tr>
<td><code>opsec_session_end_reason</code></td>
<td>returns the reason for the OPSEC session’s termination</td>
<td>page 78</td>
</tr>
<tr>
<td><code>opsec_get_session_env</code></td>
<td>returns pointer to the OPSEC session’s environment</td>
<td>page 80</td>
</tr>
<tr>
<td><code>opsec_get_sdk_version</code></td>
<td>retrieves local SDK version and patch number</td>
<td>page 96</td>
</tr>
<tr>
<td><code>opsec_get_peer_sdk_version</code></td>
<td>retrieves the peer’s SDK version and patch number</td>
<td>page 96</td>
</tr>
<tr>
<td><code>opsec_get_local_address</code></td>
<td>retrieves local IP address and port</td>
<td>page 97</td>
</tr>
<tr>
<td><code>opsec_get_peer_address</code></td>
<td>retrieves the peer’s IP address and port</td>
<td>page 98</td>
</tr>
</tbody>
</table>
Structures and Functions for Passing Information

OpsecInfo is a general purpose structure for storing and retrieving information. It is used to pass information between OPSEC entities. The specific data stored in a specific OpsecInfo structure are determined by the OPSEC entities that use them, so the OPSEC Client and Server developers must agree on the data beforehand.

The functions that handle an OpsecInfo structure are listed in the table below.

Table 1-5  OpsecInfo functions

| function name     | description                                      | See ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>opsec_info_init</td>
<td>initializes an OpsecInfo structure</td>
<td>page 81</td>
</tr>
<tr>
<td>opsec_info_destroy</td>
<td>destroys an OpsecInfo structure</td>
<td>page 82</td>
</tr>
<tr>
<td>opsec_info_set</td>
<td>sets a value in an OpsecInfo structure</td>
<td>page 82</td>
</tr>
<tr>
<td>opsec_info_get</td>
<td>retrieves a value from an OpsecInfo structure</td>
<td>page 83</td>
</tr>
</tbody>
</table>

This SESSION_OPAQUE macro can also be used for storing application specific data during a session. For more information, see “Macros” on page 105.

Main Loop, Scheduling, and Socket Event Functions

The functions that handle the main application loop and the way that it processes events are listed in the table below.

Table 1-6  Other functions

| function name         | description                                         | See ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>opsec_mainloop</td>
<td>starts the main application loop</td>
<td>page 84</td>
</tr>
<tr>
<td>opsec_schedule</td>
<td>schedules a function’s execution</td>
<td>page 85</td>
</tr>
<tr>
<td>opsec_deschedule</td>
<td>deschedules a function’s execution</td>
<td>page 86</td>
</tr>
<tr>
<td>opsec_set_socket_event</td>
<td>defines a handler function for a given socket event</td>
<td>page 87</td>
</tr>
<tr>
<td>opsec_del_socket_event</td>
<td>deletes the definition of a handler for a given socket event</td>
<td>page 88</td>
</tr>
</tbody>
</table>
**OPSEC Utilities**

OPSEC SDK Version 5.0 and higher introduces the following utilities:

- **keep alive**
  
  This service avoids TCP level timeouts by generating traffic on the TCP connection.

- **ping**
  
  This service sends small fixed data packets to the OPSEC peer, and measures the time elapsed until the peer responds.

Other OPSEC utilities enable you to set the session timeout, obtain the SDK version, and retrieve information about the local and peer entities. OPSEC utilities are listed in Table 1-7.

**Table 1-7  OPSEC utility functions**

<table>
<thead>
<tr>
<th>function name</th>
<th>description</th>
<th>See ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>opsec_start_keep_alive</td>
<td>starts the OPSEC keep-alive service, enabling OPSEC applications to avoid TCP level timeouts</td>
<td>page 90</td>
</tr>
<tr>
<td>opsec_stop_keep_alive</td>
<td>stops the OPSEC keep-alive service</td>
<td>page 90</td>
</tr>
<tr>
<td>opsec_ping_peer</td>
<td>pings the peer and measures the round-trip response time</td>
<td>page 92</td>
</tr>
<tr>
<td>opsec_set_session_timeout</td>
<td>sets the session timeout</td>
<td>page 94</td>
</tr>
<tr>
<td>opsec_set_session_timeout_handler</td>
<td>registers a the handler to be called when a session times out</td>
<td>page 95</td>
</tr>
<tr>
<td>opsec_get_sdk_version</td>
<td>retrieves local SDK version and patch number</td>
<td>page 96</td>
</tr>
<tr>
<td>opsec_get_peer_sdk_version</td>
<td>retrieves the peer’s SDK version and patch number</td>
<td>page 96</td>
</tr>
<tr>
<td>opsec_get_local_address</td>
<td>retrieves local IP address and port</td>
<td>page 97</td>
</tr>
<tr>
<td>opsec_get_peer_address</td>
<td>retrieves the peer’s IP address and port</td>
<td>page 98</td>
</tr>
<tr>
<td>opsec_get_peer_cert_hash</td>
<td>retrieves the peer’s certificate hash (if authentication method requires certificates)</td>
<td>page 103</td>
</tr>
</tbody>
</table>
SIC Utilities

The functions listed in Table 1-8 enable you to retrieve information about the SIC session.

Table 1-8 SIC utility functions

<table>
<thead>
<tr>
<th>function name</th>
<th>description</th>
<th>See ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>opsec_get_my_sic_name</td>
<td>returns the local SIC name set for the OPSEC environment</td>
<td>page 102</td>
</tr>
<tr>
<td>opsec_sic_get_peer_sic_name</td>
<td>returns the SIC name of the peer if the designated session is a SIC session</td>
<td>page 90</td>
</tr>
<tr>
<td>opsec_sic_get_sic_method</td>
<td>returns the SIC method chosen if the designated session is a SIC session</td>
<td>page 92</td>
</tr>
<tr>
<td>opsec_sic_get_peer_cert_hash</td>
<td>retrieves the certificate hash of the OPSEC session peer (on the other end of the OPSEC connection.)</td>
<td>page 103</td>
</tr>
</tbody>
</table>

Event Handlers

An OPSEC Client or Server entity responds to the following types of events:

Specifically:

Table 1-9 Event Handler Client/Server Breakdown

<table>
<thead>
<tr>
<th>Events</th>
<th>Client</th>
<th>Server</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPSEC_SESSION_START</td>
<td>yes</td>
<td>yes</td>
<td>page 12 7</td>
</tr>
<tr>
<td>OPSEC_GENERIC_SESSION_START</td>
<td>yes</td>
<td>yes</td>
<td>page 12 8</td>
</tr>
<tr>
<td>OPSEC_SESSION_END</td>
<td>yes</td>
<td>yes</td>
<td>page 12 9</td>
</tr>
</tbody>
</table>
OPSEC Data Structures and Functions

Event Descriptions

- **OPSEC_SESSION_START**: an OPSEC session has been initialized.
  - This event is triggered when an OPSEC session has been created, but before the session is completely established.

- **OPSEC_SESSION_ESTABLISHED**: an OPSEC session has been established.
  - This event is triggered when Server has acknowledged the session’s creation.

- **OPSEC_GENERIC_SESSION_START**: an OPSEC session has been started.
  - This event is triggered when an OPSEC generic session has been created, but before the session is completely established.

- **OPSEC_SESSION_END**: calls an event handler on the Server side for incomplete connections.

- **OPSEC_GENERIC_SESSION_END**: an OPSEC session has been ended.
  - This event is triggered when an OPSEC generic session has been ended.

- **OPSEC_SERVER_FAILED_CONN**: an SIC error prevented a connection from being established.

  **Note**: Not available in conjunction with the OPSEC PORT.

- Service-specific events: these events vary according to the OPSEC API (e.g. LEA, SAM), and are described in detail in the corresponding documentation (e.g. the LEA (Log Export API) Specification).
The response to each type of event is handled by the event handler (callback) functions set in the call to `opsec_init_entity` for the OPSEC entity. For example, the callbacks for `OPSEC_SESSION_START` and `OPSEC_SESSION_END` are set using the attributes listed in the table below:

<table>
<thead>
<tr>
<th>value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OPSEC_SESSION_START_HANDLER</code></td>
<td>The event handler for the <code>OPSEC_SESSION_START</code> event (see page 127).</td>
</tr>
<tr>
<td><code>OPSEC_GENERIC_SESSION_START_HANDLER</code></td>
<td>The event handler for the <code>OPSEC_GENERIC_SESSION_START</code> event (see page 128).</td>
</tr>
<tr>
<td><code>OPSEC_SESSION_ESTABLISHED_HANDLER</code></td>
<td>The event handler for the <code>OPSEC_SESSION_ESTABLISHED</code> event (see page 129).</td>
</tr>
<tr>
<td><code>OPSEC_SESSION_END_HANDLER</code></td>
<td>The event handler for the <code>OPSEC_SESSION_END</code> event (see page 130).</td>
</tr>
<tr>
<td><code>OPSEC_GENERIC_SESSION_END_HANDLER</code></td>
<td>The event handler for the <code>OPSEC_GENERIC_SESSION_END</code> event (see page 131).</td>
</tr>
<tr>
<td><code>OPSEC_SERVER_FAILED_CONN_HANDLER</code></td>
<td>The event handler for the <code>OPSEC_SERVER_FAILED_CONN</code> event (see page 132).</td>
</tr>
</tbody>
</table>

Service-specific attributes are listed in the corresponding documentation.

OPSEC entities can also respond to user-defined events, as specified using the Events API. For further details, see “Multithreaded OPSEC Applications” on page 33.
Session Multiplexing

OPSEC 5.0 and higher introduces session multiplexing. With multiplexing, one TCP connection can be used for multiple concurrent sessions. To take advantage of multiplexing, the OPSEC Client must create several concurrent sessions using the same Client and Server Entities. The first session created will establish the TCP connection. Any subsequent sessions will then use this connection. When all sessions are closed, the TCP connection will be closed as well.

The call to \texttt{opsec\_init\_entity} enables you to specify that sessions be created in one of the following ways:

- All sessions may share the same connection.
- No more than a designated number of sessions may share the same connection.
- Every session requires a separate connection.
- The level of multiplexing is determined by the OPSEC infrastructure.

For more information, see “\texttt{opsec\_init\_entity}” on page 69.

OPSEC Unique ID

An OPSEC Unique ID is a data structure containing a unique ID allowing the user to manipulate the data structure. A unique ID is a 128 bit (world-unique per generation) number. The unique ID mechanism is used by the NG based logging system as well as other places.

For more specific API information, see “OPSEC Unique IDs” on page 116.

OPSEC Values

An OPSEC value is an abstract type available for OPSEC NG and higher. OPSEC values are used by AMON (Application Monitoring API) applications.

OPSEC values enable the storage of multiple data types within a single type definition. These values can then be handled abstractly. OPSEC values can hold all value types used by OPSEC. These types are listed in Table 2-73 on page 125.
The functions listed in Table 1-11 enable you to create, duplicate, destroy, store and retrieve data in OPSEC value types.

Table 1-11  OPSEC value functions

<table>
<thead>
<tr>
<th>function name</th>
<th>description</th>
<th>See ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>opsec_value_create</td>
<td>initializes an OPSEC value</td>
<td>page 120</td>
</tr>
<tr>
<td>opsec_value_dest</td>
<td>destroys an OPSEC value</td>
<td>page 120</td>
</tr>
<tr>
<td>opsec_value_dup</td>
<td>duplicates an existing OPSEC value</td>
<td>page 121</td>
</tr>
<tr>
<td>opsec_value_set</td>
<td>sets an OPSEC value</td>
<td>page 121</td>
</tr>
<tr>
<td>opsec_value_get</td>
<td>retrieves a value from opsec_value_t</td>
<td>page 122</td>
</tr>
<tr>
<td>opsec_value_copy</td>
<td>copies a value from one OPSEC value to another</td>
<td>page 123</td>
</tr>
<tr>
<td>opsec_value_get_type</td>
<td>returns the type that is stored in opsec_value_t</td>
<td>page 123</td>
</tr>
</tbody>
</table>

**Multithreaded OPSEC Applications**

OPSEC SDK Version 4.1 and higher supports multithreaded applications. An application can spawn multiple threads, configure each as multithread-safe and then execute the opsec_mainloop function, creating multiple instances of the OPSEC event dispatching mechanism in the same process. The Events API allows multithread-safe interaction between OPSEC threads and application threads. A multithread OPSEC application must use this API to guarantee safe interthread interaction.

The Events API enables the creation of user-defined events that can be raised or triggered by any thread. These events are then processed by a thread that executes the opsec_mainloop function.

**Multithread Level**

- **Reentrant APIs** — the multithread level of most OPSEC APIs is “Reentrant”. This means that:
  - Multiple threads may use the same API concurrently.
  - Multiple threads may not share data generated by OPSEC APIs. Each thread must create its own environment, entities, sessions, etc.
- **Thread-Safe APIs** — the only “Thread-Safe” APIs are the Events APIs which use Opsecenv for inter-process communication.
• Unsafe APIs —
  • CPMI APIs are not Thread-Safe. This means, that there can only be one thread using CPMI in a process.
  • Not all authentication and encryption are Thread-Safe. A multithread application should not be configured to use them. See “connection type values” on page 73 for details.
  • opsec_errno global variable is not Thread-Safe.

Note - For Multithreaded application cleanup use opsec_DllMain, for more information, see “opsec_DllMain” on page 91

**Events API Overview**

Before starting its threads, an application creates user-defined events (by calling opsec_new_event_id) and an OPSEC environment for each thread (by calling opsec_init with OPSEC_MT attribute). Each thread registers handlers for the events it will process (by calling opsec_set_event_handler).

To raise (trigger) an event instance, a thread calls opsec_raise_event or opsec_raise_persistent_event, specifying the OPSEC environment (the thread) in which the handler for the event is defined.

If the same handler is set multiple times, either by the same thread or by different threads, its set time is updated.

Event handlers are called asynchronously. That is, they are not called immediately but rather the next time the thread containing the event handler gains focus. Then they are processed only after all pending opsec_mainloop events (scheduled and socket events, that is, not user-defined events) have been processed.

Events are uniquely identified by the ID number of their event class and the data passed to the event handler when the event is raised.

Event handlers are uniquely identified by the ID number of the event class they handle, the handler function they call, and the data set when the event handler is defined.

Note that two kinds of data are passed to an event instance:
  • the data set when the event handler is defined
• the data set when the event is raised

**Note** - The OPSEC event handlers conform to the Win32 cdecl calling convention.

See "**General Event Handlers**" on page 133 for further details on defining event handlers.

**User-Defined Events**

There are two types of user-defined events:

• **regular events**
  
  When a regular event is raised (by calling `opsec_raise_event`), all its associated handlers are called, one after another in the sequence in which they were defined (by calling `opsec_set_event_handler`). Once all the handlers have been called, the event is unraised. If the event is then raised again, it is handled in the same way.

  If there are no event handlers defined for the event, the event is lost.

• **persistent events**
  
  When a persistent event is raised (by calling `opsec_raise_persistent_event`), the event is stored. A persistent event can be raised before any handlers are defined for the event — even before an `opsec_mainloop` is executed in the target thread.

  If and when a handler is defined for the event at a later point, the handler is called with all pending persistent events. Any number of handlers can be defined, and they will be called in the sequence in which they were defined; this is true even for handlers defined after the event is raised.

  A persistent event is not unraised after it has been handled (though the application can delete it by calling `opsec_unraise_event`). Raising it again has no effect.

  Persistent events can be used to define states of logical resources.

If a regular event is raised, and before it is handled a persistent event with the same identity is raised, then the regular event is ignored and only the persistent event is handled.

If a persistent event is raised and then a regular event with the same identity is raised, the regular event is ignored and the persistent event is unaffected.

Events are processed only once `opsec_mainloop` is running.
Multithreaded OPSEC Applications

**Terminating opsec_mainloop**

_opsec_mainloop_ exits only if the following conditions are met:

- All event queues associated with the registered event handlers are empty.
- There are no handlers registered.

To allow _opsec_mainloop_ to terminate, proceed as follows:

1. For each event, delete all its handlers (using _opsec_del_event_handler_).
   
   Once all of an event’s handlers have been deleted, all its event instances are unraised.

2. Unraise all persistent events (using _opsec_unraise_event_).
   
   A persistent event remains raised even if all its event handlers have been deleted, so persistent events must be explicitly unraised.

**Events API Functions**

The functions used to implement multithreading are listed in Table 1-12.

<table>
<thead>
<tr>
<th>function name</th>
<th>description</th>
<th>see ...</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>opsec_new_event_id</em></td>
<td>creates an event class to be used in the other event functions</td>
<td>page 108</td>
</tr>
<tr>
<td><em>opsec_set_event_handler</em></td>
<td>registers an event class and its handler</td>
<td>page 109</td>
</tr>
<tr>
<td><em>opsec_del_event_handler</em></td>
<td>unregisters an event class and its handler</td>
<td>page 109</td>
</tr>
<tr>
<td><em>opsec_suspend_event_handler</em></td>
<td>suspends processing an event class</td>
<td>page 110</td>
</tr>
<tr>
<td><em>opsec_resume_event_handler</em></td>
<td>resumes processing an event class</td>
<td>page 111</td>
</tr>
<tr>
<td><em>opsec_raise_event</em></td>
<td>raises (triggers) a regular event instance</td>
<td>page 112</td>
</tr>
<tr>
<td><em>opsec_raise_persistent_event</em></td>
<td>raises (triggers) a persistent event instance</td>
<td>page 113</td>
</tr>
<tr>
<td><em>opsec_israised_event</em></td>
<td>queries the status of an event instance</td>
<td>page 114</td>
</tr>
<tr>
<td><em>opsec_unraise_event</em></td>
<td>cancels an event instance</td>
<td>page 114</td>
</tr>
</tbody>
</table>
Client-Server Connection

Using the Check Point SIC (Secure Internal Communication) infrastructure, OPSEC applications can establish one of the following types of connections with VPN-1:

- Authenticated and encrypted connection using SSL (Secure Socket Layer) — The data transferred between the OPSEC compatible application and VPN-1 is encrypted using a 3DES or RC4 key. This is done only after the LEA host is authenticated with VPN-1. An authenticated and encrypted connection is the most secure, and is highly recommended for all OPSEC applications.

- Authenticated connection— The OPSEC compatible application and VPN-1 must verify each other’s identities before any data is transferred.

Note - Data integrity is not guaranteed for authenticated connections.

- Clear connection— The OPSEC compatible application and VPN-1 can transfer data without restrictions.

The different authentication and encryption methods available for each connection type are listed below.

Table 1-13 Types of Connections

<table>
<thead>
<tr>
<th>Type of Connection</th>
<th>SIC method</th>
<th>Compressed?</th>
<th>Authentication Method</th>
<th>Encryption Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authenticated and encrypted</td>
<td>ssl</td>
<td>no</td>
<td>SSL based authentication.</td>
<td>Data is encrypted using a 3DES key.</td>
</tr>
<tr>
<td></td>
<td>ssl_opsec</td>
<td>no</td>
<td>Certificate authentication— both Client and Server must provide certificates created and signed by a trusted Certificate Authority.</td>
<td>Data is encrypted using an RC4 key.</td>
</tr>
<tr>
<td></td>
<td>sslca</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sslca_comp</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sslca_rc4</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sslca_rc4_comp</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 1-13: Types of Connections

<table>
<thead>
<tr>
<th>Type of Connection</th>
<th>SIC method</th>
<th>Compressed?</th>
<th>Authentication Method</th>
<th>Encryption Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authenticated and encrypted (Cont.)</td>
<td>asym_sslca</td>
<td>no</td>
<td>Asymmetric certificate authentication (for Client applications only)—only the Server must provide a certificate created and signed by a trusted Certificate Authority.</td>
<td>Data is encrypted using a 3DES key.</td>
</tr>
<tr>
<td></td>
<td>asym_sslca_comp</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>asym_sslca_rc4</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>asym_sslca_rc4 _comp</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authenticated</td>
<td>ssl_clear</td>
<td>N/A</td>
<td>SSL based authentication. This is the recommended method for authentication when data encryption is not required.</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>ssl_clear_opsec</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fwn1</td>
<td></td>
<td>Check Point’s proprietary authentication algorithm (for backward compatibility).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>auth_opsec</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sslca_clear</td>
<td>no</td>
<td>Certificate authentication—both Client and Server must provide certificates created and signed by a trusted Certificate Authority.</td>
<td></td>
</tr>
<tr>
<td>Local</td>
<td>local</td>
<td>no</td>
<td>Based on a secret on the local disk accessible to both client and server</td>
<td></td>
</tr>
<tr>
<td>Clear</td>
<td>none</td>
<td>N/A</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
To implement authentication or encryption, follow these steps:

1. If the XXX and VPN-1 are to be authenticated using Check Point's proprietary authentication algorithm (fwn1, auth_opsec) or SSL based authentication (ssl, ssl_opsec, ssl_clear_opsec), establish an authentication key for communication between the machines. The machines will then identify themselves using the authentication key.

   For instructions, see “Establishing an authentication key” on page 40 of the OPSEC API Specification.

2. If the XXX and VPN-1 are to be authenticated using certificates (sslca, sslca_comp, sslca_rc4, sslca_rc4_comp, asym_sslca, asym_sslca_comp, asym_sslca_rc4, asym_sslca_rc4_comp), obtain the certificates in PKCS#12 format from the Certificate Authority.

   For instructions, see “Pulling Certificates” on page 17 of the OPSEC API Specification.

3. Enable the use of the SIC infrastructure.

   For instructions, see “Enabling SIC” on page 18 of the OPSEC API Specification.

4. Configure VPN-1 to use the appropriate connection method.

   For instructions, see the Security Administration Guide.

5. Configure the OPSEC application to use the appropriate connection method.

   For instructions, see “Configuring the OPSEC Application” on page 44.
Establishing an authentication key

For the purpose of establishing an authentication key, it does not matter whether a given machine is running as an OPSEC Server or an OPSEC Client. However, you must configure the machine running the VPN-1 first.

Note - If the Client and Server are both on the same machine, this procedure is unnecessary.

To establish an authentication key, follow these steps:

1. On the machine running VPN-1, enter one of the following commands at the command line:
   - For an SSL based connection (authenticated or authenticated and encrypted), enter:
     ```
     fw putkey -opsec -ssl opsec_host
     ```
   - For a backward compatible authenticated connection using the Check Point proprietary authentication algorithm, enter:
     ```
     fw putkey -opsec opsec_host
     ```
   *opsec_host* is the resolvable name or IP address of the machine running the OPSEC application.
   You will be prompted to enter the authentication key, which must be at least 6 characters long.

2. On the OPSEC host, enter one of the following commands at the command line:
   - For an SSL based (authenticated or authenticated and encrypted) connection, enter:
     ```
     opsec_putkey -port fw -ssl fw1_host
     ```
   - For a backward compatible authenticated connection using the Check Point proprietary authentication algorithm, enter:
     ```
     opsec_putkey -port fw fw1_host
     ```
   *fw1_host* is the resolvable name or IP address of the machine running VPN-1.
   You will be prompted to enter the authentication key. Enter the same key you entered in step 1.

*opsec_putkey* is an executable that is part of the OPSEC SDK.

Usage: opsec_putkey [-ssl] [-port (port_number | fw)] [-p pswd] host
The opsec_putkey parameters are explained in the table below.

<table>
<thead>
<tr>
<th>parameter</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-ssl switch</td>
<td>opsec_putkey will exchange SSL only keys. If this is not supplied, opsec_putkey will exchange FWN1 only keys</td>
</tr>
<tr>
<td>-port &lt;port number&gt;</td>
<td>opsec_putkey will exchange keys by connecting to the VPN-1 machine using port &lt;port number&gt;</td>
</tr>
<tr>
<td>-port fw</td>
<td>opsec_putkey will exchange keys by connecting to the VPN-1 machine using the default key exchange port. If not specified, opsec_putkey will only create the keys in the local key file and will not exchange them with the VPN-1 machine</td>
</tr>
<tr>
<td>-p psswd</td>
<td>opsec_putkey will use the provided password will not query for a password. If not specified, opsec_putkey will query for a password</td>
</tr>
<tr>
<td>host</td>
<td>This mandatory parameter specifies the VPN-1</td>
</tr>
</tbody>
</table>

The key is saved in the OPSECDIR directory on the machine running the OPSEC application (see “OPSECDIR” on page 23). If OPSECDIR environment variable is not defined, the key is saved in the current directory.

If several machines will be communicating with each other then you must follow this procedure for each machine, possibly using a different key for each pair of machines. For example, if a machine named fog is communicating with machines named olive and grape, then fog-olive connections can use one key and fog-grape connections can use another key.

**Pulling Certificates**

If the OPSEC application and VPN-1 are to be authenticated using certificates, you must obtain the certificates in PKCS#12 format from the VPN-1 Certificate Authority.

For certificate authentication, you must pull the certificates for both Client and Server applications. For asymmetric certificate authentication, you must obtain the certificate for the Server application only.
Either the location, or the content of the OPSEC application’s certificate must be specified to opsec_init using the OPSEC_SSLCA_FILE attribute, or OPSEC_SSLCA_BUFFER attribute, for each respective option. For more information, see “opsec_init” on page 60.

To pull a certificate, type the following at the command line on the machine running the application that needs the certificate (recall that for asym_ssl, the Client need not pull the certificate):

```
    opsec_pull_cert -h host -n object_name -p password [-o cert_file]
    [-od dn_file]
```

opsec_pull_cert is an executable that is an addition to the OPSEC SDK. Its parameters are explained in the table below.

**Table 1-15** opsec_pull_cert parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h host</td>
<td>The resolvable name or IP address (in dot format) of SmartCenter Server running the Certificate Authority.</td>
</tr>
<tr>
<td>-n object_name</td>
<td>The VPN-1 name of the object for which the certificate is being requested.</td>
</tr>
<tr>
<td>-p password</td>
<td>The one-time password entered in the SmartDashboard when defining the network object.</td>
</tr>
<tr>
<td>-o cert_file</td>
<td>The name of the output file (a .p12 extension is added automatically). The default file name is opsec.p12.</td>
</tr>
<tr>
<td>-od dn_file</td>
<td>The name of the file containing the SIC name of the object. If not specified, the object’s SIC name is printed to stdout.</td>
</tr>
</tbody>
</table>

You may only use opsec_pull_cert once for each application. If opsec_pull_cert fails, or if you need to run opsec_pull_cert again for any reason, you must revoke the certificate before generating a new one.

For more information about the VPN-1 Certificate Authority, see Check Point Virtual Private Networks.
Enabling SIC

OPSEC SDK Versions which are NG based use the SIC (Secure Internal Communications) framework for authenticating and encrypting connections.

SIC uses a set of rules to determine the types of connections that the OPSEC Server will accept. These rules are created dynamically at runtime based on the OPSEC application’s configuration information. Alternately, these rules may be specified in the SIC policy file. For more information about the SIC policy file, see “SIC Policy File” on page 50.

For enabling SIC with different CA’s, see “Multi-SIC Configuration” on page 55.

To enable SIC, follow these steps:

1. Define the OPSEC application’s SIC name.
   
   In SIC connections, the OPSEC application is identified by its SIC name both as a Client and as a Server. The SIC name is the application’s full DN (distinguished name) as defined by the VPN-1 SmartCenter Server. The SIC name may be specified using the \texttt{OPSEC\_SIC\_NAME} attribute of \texttt{opsec\_init} (see “opsec\_init” on page 60), or alternately by adding the line below the OPSEC application’s configuration file:

   \begin{verbatim}
   opsec\_sic\_name SIC\_name
   \end{verbatim}

   SIC\_name is the OPSEC application’s SIC name. If the SIC name contains white spaces, it must be delimited by quotation marks. For example, below are two valid ways to specify SIC names in the configuration file:

   \begin{verbatim}
   opsec\_sic\_name cn=elvis,o=graceland
   opsec\_sic\_name “cn=elvis, o=graceland”
   \end{verbatim}

   \textbf{Note} - These instructions apply to configuring a single-SIC identity (for connecting with modules from the same SmartCenter). To configure the OPSEC application with multiple SIC identities, see “Multi-SIC” on page 55.

2. Define the SIC name of the OPSEC Server entity.
   
   The Server’s SIC name must be defined in:
   
   - the OPSEC Server application

   The Server’s SIC name is the same as the name defined in step 1.
Specifying the Server’s SIC name a second time enables the dynamic generation of SIC rules. If a SIC policy file is provided, supplying the Server’s SIC name is optional.

- the OPSEC Client application

In addition to enabling dynamic SIC rule generation, the OPSEC Client uses the Server entity’s SIC name to verify that it connected to the correct SmartCenter Server or VPN-1 Module. If a SIC policy file is provided, supplying the Server’s SIC name is optional.

The Server entity’s SIC name may be specified using the OPSEC_ENTITY_SIC_NAME attribute of opsec_init_entity (see “opsec_init_entity” on page 69), or alternately by adding the line below the OPSEC application’s configuration file:

```
server_name opsec_entity_sic_name SIC_name
```

server_name is the name of the OPSEC Server entity as defined in the application. SIC_name is the Server entity’s SIC name.

### Configuring the OPSEC Application

You can configure an OPSEC application in several ways:

- by coding the information into the application
- by using a configuration file (see page 45)
- by using the command line (see page 49)
- by using a SIC policy file (see page 50)

See also “Precedence” on page 54.

Configuration information is stored in the application’s OpsecEnv structure along with other information about the OPSEC environment. For more information, see “Environment Structure and Functions” on page 25.

Configuration information can be retrieved using the opsec_get_conf function. See “opsec_get_conf” on page 67.
Configuration Files

Each OPSEC application can have one or more configuration files associated with it as part of its environment.

You can specify the configuration file or files to be used by using the OPSEC_CONF_FILE attribute of opsec_init. For more information, see “opsec_init” on page 60.

Configuration file syntax is described in Table 1-16 on page 45. server_name is the name of the OPSEC Server entity as defined in the application. See “opsec_init_entity” on page 69.

<table>
<thead>
<tr>
<th>this syntax...</th>
<th>defines...</th>
</tr>
</thead>
<tbody>
<tr>
<td>server_name ip ip_address</td>
<td>...the IP address of the initialized OPSEC entity. The IP address is in dot format.</td>
</tr>
<tr>
<td>server_name host host_name</td>
<td>...the resolvable host name of the initialized OPSEC entity.</td>
</tr>
<tr>
<td>server_name port port_number</td>
<td>...the port number to be used for establishing a clear connection with the initialized OPSEC entity. This connection does not use the SIC framework.</td>
</tr>
<tr>
<td>server_name auth_port port_number</td>
<td>...the port number to be used for establishing SIC based connections with the initialized OPSEC entity.</td>
</tr>
<tr>
<td>server_name auth_type method</td>
<td>...the types of connections allowed to auth_port. method may be one of: ssl, ssl_opsec, sslca, sslca_comp, sslca_rc4, sslca_rc4_comp, asym_sslca, asym_sslca_comp, asym_sslca_rc4, asym_sslca_rc4_comp, ssl_clear_opsec, fwn1, auth_opsec, local and none. For details, see “Client-Server Connection” on page 37.</td>
</tr>
<tr>
<td>server_name no_nagle</td>
<td>...Disable Nagle's algorithm for the server's connections.</td>
</tr>
<tr>
<td>server_name conn_buf_size size</td>
<td>The maximum buffer size (in bytes) for the entity's connections. If not specified default is 1 Megabyte.</td>
</tr>
</tbody>
</table>
The type of connection defined for an OPSEC Client must be compatible with the type of connection defined for the OPSEC Server.

The following is a fragment of an OPSEC Server configuration file. It configures the CVP Server named my_cvp to listen on port number 18181. The Server's host name is elvis.

```
my_cvp port 18181
my_cvp host elvis
```

Configuration file parameters may contain white space if they are delimited by quotation marks. However, keywords may not be delimited by quotation marks. For example, the following syntax is valid:

```
my_cvp port "18181"
```

But the following syntax is not valid:

```
my_cvp "port" 18181
```
For additional examples and configuration instructions for OPSEC Clients and Servers, see “Configuring the OPSEC Server application” below and “Configuring the OPSEC Client application” on page 49.

**Configuring the OPSEC Server application**

To configure an OPSEC Application as a Server, define its connection with OPSEC Clients in the configuration file using the following syntax:

- For a clear connection with a Client running VPN-1/FireWall-1 Version 4.1.2 or lower, use:
  
  ```
  server_name port port_number
  ```

- For all other types of connections, use:
  
  ```
  server_name auth_port port_number
  server_name auth_type method
  ```

You can bind the Server to a given IP address by adding the following line to the configuration file:

```
server_name ip ip_address
```

If you do not specify the ip parameter, then the Server binds to all of the computer's interfaces.

If you specify an IP address that does not exist, the bind will fail.

If you specify the loopback interface (127.0.0.1), you will only be able to access the computer from itself.
The configuration parameters are explained in the table below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>server_name</td>
<td>The name of OPSEC Server entity as defined in the application. See “opsec_init_entity” on page 69.</td>
</tr>
<tr>
<td>port_number</td>
<td>The port number.</td>
</tr>
<tr>
<td>method</td>
<td>One of the following:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>NG based only</th>
<th>Authentication method</th>
<th>Encryption method</th>
<th>Compressed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssl</td>
<td>*</td>
<td>SSL based</td>
<td>3DES</td>
<td>no</td>
</tr>
<tr>
<td>ssl_opsec</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sslca</td>
<td>*</td>
<td>certificates</td>
<td>3DES</td>
<td>no</td>
</tr>
<tr>
<td>sslca_clear</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sslca_comp</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sslca_rc4</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sslca_rc4_comp</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>asym_sslca</td>
<td>*</td>
<td>asymmetric certificates</td>
<td>3DES</td>
<td>no</td>
</tr>
<tr>
<td>asym_sslca_comp</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>asym_sslca_rc4_comp</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssl_clear</td>
<td>*</td>
<td>SSL based</td>
<td>none</td>
<td>no</td>
</tr>
<tr>
<td>ssl_clear_opsec</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fwn1</td>
<td>*</td>
<td>Check Point proprietary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>auth_opsec</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example

my_ufp ip 133.45.67.102
my_ufp auth_port 18182
my_ufp auth_type ssl_clear_opsec
This means that Server my_ufp is bound to IP address 133.45.67.102 and communicates with its Clients using port 18182. The connection between the Server and its Clients is authenticated. The Server can communicate with Clients running VPN-1 Versions that are NG based as well as Clients running VPN-1/FireWall-1 Version 4.1.2 or lower.

**Configuring the OPSEC Client application**

To configure an OPSEC Application as a Client, define its connection with the OPSEC Server in the configuration file using the following syntax:

- For a clear connection that is not SIC based (for example, with a Server running VPN-1/FireWall-1 Version 4.1.2 or lower), use:

  ```
  server_name ip ip_address
  server_name port port_number
  ```

- For all SIC based connections, use:

  ```
  server_name ip ip_address
  server_name auth_port port_number
  server_name auth_type method
  ```

The parameters are explained in table Table 1-17.

**Example**

```
lea_server ip 143.193.22.5
lea_server auth_port 18184
lea_server auth_type fwn1
```

This means that the OPSEC application is configured as a LEA Client that communicates on port 18184 with the LEA Server located at IP address 143.193.22.5. The connection between Server and Client is authenticated using Check Point's authentication algorithm.

**Command line**

You also can specify configuration information via the command line. To do so, your application must call opsec_init using the OPSEC_CONF_ARGV attribute. For more information, see “opsec_init” on page 60.

**Note** - SIC names cannot be specified via the command line.
The -v switch signals your application that you are passing in configuration information. The switch can be used multiple times to pass in multiple argument strings. For example, you can invoke your application as follows:

```
myapp -v cvp_server auth_port 18181 -v lea_server host elvis
```

That is, -v argument_string is the equivalent of a line in the configuration file.

**Duplicate values**

If there are duplicate attributes or paths in the argument strings, then the left-most one overrides those on the right. For example, suppose the application is invoked with the following command:

```
myapp -v cvp_server port 18182 -v cvp_server port 18181
```

In this case, the port used to communicate with the CVP Server will be set to 18182.

**Order of command line arguments**

Command line arguments that are not meant to be passed to the OPSEC environment should appear before those that are meant to be passed to the OPSEC environment. In other words, the -v switches should be the rightmost switches.

Suppose -x and -y are application specific command line arguments. It is fine to invoke your application as follows:

```
myapp -x -y -v cvp_server auth_port 18181
```

However, if you invoke your application with the arguments listed in the following order, it may be complicated to retrieve the -y argument.

```
myapp -x -v cvp_server auth_port 18181 -y
```

For more information, see the example illustrating opsec_init page 60 on page 60.

**SIC Policy File**

Each OPSEC application can have a SIC policy file associated with it as part of its environment. This file contains a set of rules specifying the different types of connections that may be established between the OPSEC application and VPN-1. These rules may describe inbound connections (for OPSEC Server applications) or outbound connections (for OPSEC Client applications).

See also “SIC Policy and Precedence” on page 54.
SIC Policy Syntax

The SIC policy file has the following structure:

```plaintext
#!Format 2.0
# a list of rules for inbound connections
[Inbound rules]
rule 1
rule 2
...
rule n

# a list of rules for outbound connections
[Outbound rules]
rule 1
rule 2
...
rule m
```

The “#!Format 2.0” line is mandatory. Inbound rules are indicated by the line “[Inbound rules]”. Outbound rules are indicated by the line “[Outbound rules]”. Comments are indicated by “#”.

Each rule in the SIC policy file describes a connection attempt (as defined by the OPSEC application host, the VPN-1 host, the port number, and service) and one or more connection methods. A SIC rule consists of five columns delimited by “;” as follows:

```plaintext
apply_to ; peer ; port ; service ; method
```
Table 1-18 lists the possible values for each of these columns.

**Table 1-18  SIC policy rule columns**

<table>
<thead>
<tr>
<th>Apply to...</th>
<th>Peer</th>
<th>Port</th>
<th>Service</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>The OPSEC host:</td>
<td>• ME</td>
<td>• ANY</td>
<td>The OPSEC service:</td>
<td>• ANY</td>
</tr>
<tr>
<td></td>
<td>• ANY</td>
<td>• One or more</td>
<td>If VPN-1 Pro/Express uses an earlier version of the OPSEC SDK, one or more of:</td>
<td>One or more of: none, ssl, sslca, sslca_comp, sslca_rc4, sslca_rc4_comp, asym_sslca, asym_sslca_comp, asym_sslca_rc4, asym_sslca_rc4_comp, fwn1, sslclear, local</td>
</tr>
<tr>
<td></td>
<td>The OPSEC host's SIC name</td>
<td>port number:</td>
<td>fwn1_opsec, ssl_clear_opsec, fwn1_opsec.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(can be common name only).</td>
<td>• ANY</td>
<td>• If VPN-1 uses OPSEC SDK Version NG based, see Table 1-19.</td>
<td></td>
</tr>
</tbody>
</table>

**Table 1-19  OPSEC Services**

<table>
<thead>
<tr>
<th>SIC policy file parameter</th>
<th>Corresponding OPSEC Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>amon</td>
<td>Application Monitoring</td>
</tr>
<tr>
<td>cpmi</td>
<td>Check Point Management Interface</td>
</tr>
<tr>
<td>cvp</td>
<td>Content Vectoring Protocol</td>
</tr>
<tr>
<td>ela</td>
<td>Event Logging</td>
</tr>
<tr>
<td>lea</td>
<td>Log Exporting</td>
</tr>
<tr>
<td>sam</td>
<td>Suspicious Activity Monitoring</td>
</tr>
<tr>
<td>uaa</td>
<td>UserAuthority</td>
</tr>
<tr>
<td>ufp</td>
<td>URL Filtering Protocol</td>
</tr>
</tbody>
</table>
ME designates the OPSEC application. ANY designates all valid values for the column (e.g. all valid hosts, port numbers, or services). A column may contain multiple values delimited by commas. If SIC names are specified using common names (cn) only, it is assumed that the domain of the SIC name is the same as the domain of the OPSEC application’s SIC name as defined in the configuration file.

The order of the rules in the SIC file corresponds to the order in which they are applied. For each connection attempt, the SIC policy is scanned until a match is found for the connection’s designated source, destination, port number, and service.

If a match is found, the list of corresponding connection methods is compared with the list of methods available to VPN-1. If the OPSEC application and VPN-1 are able to agree on a common method, the connection attempt is allowed to proceed. Note the following:

- The ordering of connection methods within a rule is important. The list of methods is scanned from left to right until a method common to both Client and Server is found.
- Even when a connection method is agreed upon, this does not guarantee that the connection attempt will be successful since the OPSEC host and the VPN-1 host must still authenticate themselves.

The SIC policy follows the principle “That Which Is Not Expressly Permitted is Prohibited.” To enforce this principle, the SIC policy implicitly adds a rule at the end of the file that drops all communication attempts not described by the other rules.

**Example**

Consider the following SIC policy fragment:

```
[Inbound rules]
# apply_to peer                       port  service method
# ----------------------------------------------------------------
ME ; “cn=elvis, o=graceland, c=us” ; 18181 ; cvp ; fwn1

[Outbound rules]
# apply_to peer                       port  service method
# ----------------------------------------------------------------
ANY ; “cn=priscilla, o=graceland, c=us” ; 18187 ; ela ; sslca, ssl
```

The inbound rule indicates that the OPSEC host will accept CVP connections on port 18181 from the VPN-1 host elvis only if VPN-1 can authenticate itself using Check Point’s authentication algorithm.
Configuring the OPSEC Application

The outbound rule indicates that the OPSEC host will initiate ELA connections on port 18187 of the VPN-1 host priscilla only if the VPN-1 ELA Server agrees to authentication and encryption using either the sslca or ssl methods.

**Precedence**

The configuration specified in the configuration file overrides the coded configuration. The configuration specified using the command line overrides both the coded configuration and the configuration specified in the configuration file.

For example, suppose your application is coded to use port 18181 for clear connections (connections that are neither authenticated or encrypted), and in the configuration file or the command line you specify that it should use port 18182 for clear connections, then your application will use port 18182.

Now suppose you specify in your configuration file that your application should use port 18181 for clear connections. If you invoke your application from the command line by defining the authenticated port as 18182, then both port types will now be defined (clear will be defined to 18181 and authenticated to 18182). That is, the port configuration parameter (18181) will not be overridden. If you are configuring an OPSEC Client, this results in undefined behavior.

**SIC Policy and Precedence**

Rules are added dynamically to the top of the runtime image of the SIC policy when either of the following is true:

- auth_type is specified in the configuration file (see page 45)
- OPSEC_SERVER_AUTH_TYPE is specified in the call to opsec_init_entity for the Server entity (see page 69).

In other words, when authentication type is specified, the configuration file and the coded configuration both override the SIC Policy file configuration.
**Multi-SIC**

In most cases, SIC is used for internal connections between different entities of the same domain (e.g. SmartCenter). Such entities use certificates generated by the internal CA, and therefore cannot communicate with entities which are managed by another SmartCenter. In some cases, however, OPSEC applications may require to communicate with peers from different domains.

One way to achieve this is by running a different thread for each domain. This complicates the application design and will work only for services which are thread safe (will not work for CPMI for example). Another, recommended way to achieve this is to use the Multi-SIC identities infrastructure. Using Multi-SIC allows an OPSEC application to configure multiple SIC identities from different domains and communicate to peer entities from these domains, from a single thread.

For using the Multi-SIC infrastructure, the OPSEC application only needs to initialize the different SIC identities. The decision of which SIC identity to use for every connection is done by the OPSEC. If the Multi-SIC application is a client trying to connect to a SIC server, the Multi-SIC infrastructure uses server's SIC name to choose which one of the client's SIC identities should be used for the connection.

For example if the server's SIC name is `cn=cp_mgmt,O=jerusalem.firstdomain.com.dowhr2`, the client will choose an identity from the same domain such as `CN=ela_client,O=jerusalem.firstdomain.com.dowhr2`. In a multi-SIC server application, the Multi-SIC infrastructure will choose the server's SIC identity according to the domain name of the SIC name of the incoming connection.

For an example of a Multi-SIC application see examples/sic directory in the OPSEC SDK 6.0 package.

**Multi-SIC Configuration**

Multi-SIC initialization is performed by calling `opsec_init_sic_id` for each SIC identity. Initialization is similar to initialization of single-SIC identity with `opsec_init`, and can be done in one of the following ways:

**Passing Arguments to opsec_init_sic_id**

A SIC identity can be initialized by calling `opsec_init_sic_id` and passing it all SIC initialization arguments, such as a SIC name and SIC SSLCA certificate file. See additional information in API documentation of “`opsec_init_sic_id`” on page 65.
**Command-line Arguments**

For initializing SIC identities with command-line see OPSEC application configuration from command line on page 49.

Note that SIC initialization arguments such as opsec_sic_name and opsec_sslca_file must be preceded by a unique name that identifies the SIC identity from other SIC identities in the command-line, for example:

```
myapp -v jerusalem opsec_sic_name
cn=cp_mgmt,O=jerusalem.firstdomain.com.dowhr2 -v jerusalem
opsec_sslca_file jerusalem opsec_sslca_file
ela_client_jerusalem.p12 -v london opsec_sic_name
CN=ela_client_london,O=london.seconddomain.com.uery7x -v london
opsec_sslca_file ela_client_london.p12
```

In this example jerusalem and london are 2 SIC identity names which separate arguments for configuring the different SIC identities.

Then the application should call opsec_init_sic_id for each of the 2 identities by specifying the unique name in each call. For example:

```
if (opsec_init_sic_id(env,OPSEC_SIC_ID_NAME,"jerusalem",OPSEC_EOL)) {
    printf("Failed to create SIC identity for jerusalem\n");
    clean_env(env, client, server1,server2);
    exit(1);
}
if (opsec_init_sic_id(env,OPSEC_SIC_ID_NAME,"london",OPSEC_EOL)) {
    printf("Failed to create SIC identity for london\n");
    clean_env(env, client, server1,server2);
    exit(1);
}
```

**Configuration File**

Initialization with a configuration file is done by:

- Adding all SIC identity configuration arguments to the configuration file.
- Initializing OPSEC (with opsec_init) with a sic configuration file
- Calling opsec_init_sic_id with a unique name that identifies the SIC identity in the configuration file.

SIC configuration file arguments are described on page 45.
For example:

```
# Configuration of first SIC identity (jerusalem)
jerusalem opsec_sic_name
CN=ela_client,O=jerusalem.firstdomain.com.dowhr2
jerusalem opsec_sslca_file ela_client_jerusalem.p12

# Configuration of second SIC identity (london)
london opsec_sic_name
CN=ela_client_london,O=seconddomain.com.uery7x
london opsec_sslca_file ela_client_london.p12
```

Note that SIC initialization arguments such as opsec_sic_name and opsec_sslca_file are preceded by a unique name that identifies the SIC identity from other SIC identities in the configuration file.

**Notes on Multi-SIC**

- When using 'local' authentication method in a Multi-SIC application, the same shared local path is used by all SIC identities. This means that the OPSEC_SHARED_LOCAL_PATH should be supplied only to opsec_init (not to opsec_init_sic_id), and that the configuration file and command-line argument opsec_shared_local_path must not be preceded with a unique SIC identity name.

- When using an external SIC policy file in a Multi-SIC applications, same SIC policy file is used by all SIC identities. This means that the OPSEC_SIC_POLICY_FILE should be supplied only to opsec_init (not to opsec_init_sic_id), and that the configuration file and command-line argument opsec_sic_policy_file must not be preceded with a unique SIC identity name.

   It also means that the shared SIC policy file should include SIC rules for all SIC identities.

- When the OPSEC application is a Multi-SIC client, it must specify the server's SIC name when initializing the server entity. Otherwise the Multi-SIC infrastructure will not be able to choose which of the client's SIC identities should be used when connecting to this server.
Platforms Compatibility

OPSEC applications based on OPSEC SDK 6.0 are supported on the same platforms and configurations as VPN-1 NGX R60. Client-server connections and operations are platform independent. This means an OPSEC application running on a certain platform can communicate and operate with a VPN-1 module running on any platform.
Chapter 2

OPSEC API (Application Programming Interface)

In This Chapter

General OPSEC Functions  page 60
Macros  page 105
Error Handling Functions  page 106
Events API Functions  page 108
OPSEC Unique IDs  page 116
OPSEC Value API  page 120
OPSEC Value Types  page 125
Event Handlers  page 127
General OPSEC Functions

This section describes each of the common OPSEC API functions in detail. The function prototypes are defined in the `opsec.h` file.

Managing Environments

The following functions initialize, destroy, and retrieve configuration information from the OPSEC environment.

```
  opsec_init
  opsec_env_destroy
  opsec_get_conf
```

**opsec_init**

`opsec_init` initializes the OPSEC environment (the `OpsecEnv` structure) for the application.

Once the environment is initialized, any number of OPSEC entities may be created within it using the `opsec_init_entity` function (see “opsec_init_entity” on page 69).

**Prototype**

```
OpsecEnv *opsec_init (...);
```
Arguments

Table 2-1  opsec_init arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>a list of attributes and their values</td>
<td>One of a variable number of attribute-value pairs. The OPSEC_EOL attribute indicates. The attributes are listed in Table 2-2 below.</td>
</tr>
</tbody>
</table>

Table 2-2  attributes and values

<table>
<thead>
<tr>
<th>attribute</th>
<th>C type</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>The following attributes are required.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPSEC_SIC_NAME</td>
<td>char *</td>
<td>The full DN (distinguished name) of the OPSEC application. For more information, see “Enabling SIC” on page 43.</td>
</tr>
<tr>
<td>The following attributes are optional and, unless</td>
<td></td>
<td></td>
</tr>
<tr>
<td>otherwise stated, may have zero or more values.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPSEC_CONF_FILE</td>
<td>char *</td>
<td>The name of the configuration file. If there is more than one OPSEC_CONF_FILE attribute, then the contents of all files listed are used in the configuration. For more information, see “Configuration Files” on page 45.</td>
</tr>
<tr>
<td>OPSEC_CONF_ARGV</td>
<td>int*,</td>
<td>The command line arguments. For more information, see “Command line” on page 49.</td>
</tr>
<tr>
<td></td>
<td>char **</td>
<td></td>
</tr>
<tr>
<td>OPSEC_SIC_POLICY_FILE</td>
<td>char *</td>
<td>The name of the SIC policy file. Only one policy file may be designated. For more information, see “SIC Policy File” on page 50.</td>
</tr>
<tr>
<td>OPSEC_SSLCA_FILE</td>
<td>char *</td>
<td>The name of the file containing the OPSEC application’s certificate. The presence of this attribute enables certificate authentication (sslca, sslca_rc4, etc.) For more information, see “Pulling Certificates” on page 41.</td>
</tr>
</tbody>
</table>
### Table 2-2  attributes and values

<table>
<thead>
<tr>
<th>attribute</th>
<th>C type value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPSEC_SSLCA_BUFFER</td>
<td>char *, size_t</td>
<td>A pointer to a buffer followed by the buffer size. The buffer holds the content of the OPSEC certificate file (see OPSEC_SSLCA_FILE attribute)</td>
</tr>
<tr>
<td>OPSEC_SHARED_LOCAL_PATH</td>
<td>char *</td>
<td>The path on the local disk where the secret for the &quot;local&quot; authentication method is placed. The path should be set to $CPDIR/database (in Unix based machines) or %CPDIR%\database (on Windows based machines). Note that the value of CPDIR environment variable may change in CP product upgrades. A change in CPDIR requires updating the OPSEC application configuration.</td>
</tr>
<tr>
<td>OPSEC_MT</td>
<td>no value</td>
<td>This indicates that the created environment is multithread-safe. In multithread applications, where more than one environment may be used concurrently, this attribute must be supplied to all opsec_init calls</td>
</tr>
</tbody>
</table>
Return Values

Pointer to an OpsecEnv structure if the environment has been initialized. NULL if the OPSEC environment has not been initialized. In this case the application should not call any OPSEC API functions. The OpsecEnv structure must not be reused. It should be destroyed after opsec_mainloop returns.
Example

Consider the following code fragment:

```c
main(int argc, char **argv)
{
  .
  .
  .
  opsec_init (OPSEC_CONF_FILE, "abc.conf",
              OPSEC_CONF_FILE, "xyz.conf",
              OPSEC_CONF_ARGV, &argc, argv,
              OPSEC_EOL);
  .
  .
}
```

`opsec_init` adds the contents of both `abc.conf` and `xyz.conf` to the `OpsecEnv` structure.

Now suppose this application is invoked with the following command:

```
myapp -x -y -v Bob Alice -v Bugs Elmer -v London Paris Madrid
```

`myapp`’s main function is called with `argc = 13`. `opsec_init` scans `argv` from right to left looking for the `-v` switches. For every “-v” that it finds, it adds the corresponding string to the `OpsecEnv` structure. The number of strings (including all the “-v” strings) is then subtracted from `argc`. For this reason it is important that the `-v` switches be the rightmost switches.

In this example, `opsec_init` sets `argc` to 3, and stores the strings “Bob Alice”, “Bug Elmer” and “London Paris Madrid” in the `OpsecEnv` structure.

**Note** - `opsec_init` does not override the default OS handler of SIGPIPE. Therefore, applications running on Unix platforms should ignore the SIGPIPE signal. This will prevent segmentation faults that may occur during load as a result of a broken-pipe.

**opsec_env_destroy**

`opsec_env_destroy` destroys an OPSEC environment.

**Prototype**

```c
void opsec_env_destroy (OpsecEnv *env);
```
Managing Environments

Chapter 2 OPSEC API (Application Programming Interface)

Arguments

Table 2-3  
opsec_env_destroy arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer (returned by opsec_init) to the OPSEC environment which will be freed</td>
</tr>
</tbody>
</table>

Return Values

None.

opsec_init_sic_id

opsec_init_sic_id should be used only in Multi-SIC applications. It is used to initialize each of the SIC identities. Most SIC initialization arguments to this function are the same as the SIC initialization arguments of opsec_init.

Prototype

int opsec_init_sic_id(OpsecEnv* env, int, ...);
Managing Environments

Arguments

Table 2-4  opsec_init_sic_id arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer (returned by opsec_init) to the OPSEC environment</td>
</tr>
</tbody>
</table>

Further attributes are listed in Table 2-5.

Table 2-5  attributes and values

<table>
<thead>
<tr>
<th>attribute</th>
<th>C type value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPSEC_SIC_ID_NAME</td>
<td>char *</td>
<td>A unique name that identifies this SIC identity from others in the Multi-SIC application. This argument is required only when configuring the SIC identities from command-line or configuration file. For further information see Multi-SIC configuration on page 55</td>
</tr>
<tr>
<td>OPSEC_SIC_NAME</td>
<td>char *</td>
<td>The full DN (distinguished name) of the SIC identity.</td>
</tr>
<tr>
<td>OPSEC_SSLCA_FILE</td>
<td>char *</td>
<td>The name of the file containing the SIC identity's certificate.</td>
</tr>
<tr>
<td>OPSEC_SSLCA_BUFFER</td>
<td>char *, size_t</td>
<td>A pointer to a buffer followed by the buffer size. The buffer holds the content of the OPSEC certificate file for the SIC identity.</td>
</tr>
<tr>
<td>OPSEC_EOL</td>
<td>no value</td>
<td>This indicates the end of the attribute-value list.</td>
</tr>
</tbody>
</table>

Return Values

0 if successful. -1 if otherwise

Note - This API must be called before any OPSEC entities are created (with opsec_init_entity).
**opsec_destroy_sic_id**

`opsec_destroy_sic_id` should be used only in Multi-SIC applications. It is used to destroy SIC identities created with `opsec_init_sic_id`.

**Note** - This API is not required in most cases since all SIC identities are implicitly destroyed when the OPSEC environment is destroyed (with `opsec_env_destroy`).

**Prototype**

```c
int opsec_destroy_sic_id (OpsecEnv* env, int attr, ...)
```

**Arguments**

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer (returned by <code>opsec_init</code>) to the OPSEC environment</td>
</tr>
</tbody>
</table>

Further attributes are listed in Table 2-7.

**Table 2-6**  
`opsec_destroy_sic_id` arguments

<table>
<thead>
<tr>
<th>attribute</th>
<th>C type value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPSEC_SIC_ID_NAME</td>
<td>char *</td>
<td>The same name that was used to create the SIC identity with <code>opsec_init_sic_id</code>.</td>
</tr>
<tr>
<td>OPSEC_SIC_NAME</td>
<td>char *</td>
<td>The full DN (distinguished name) of the SIC identity.</td>
</tr>
<tr>
<td>OPSEC_EOL</td>
<td>no value</td>
<td>This indicates the end of the attribute-value list.</td>
</tr>
</tbody>
</table>

**Return Values**

*0 if successful. -1 if otherwise.*

**opsec_get_conf**

`opsec_get_conf` retrieves configuration information from the `OpsecEnv` structure.
Configuration information is stored as a directory-like structure in OpsecEnv. `opsec_get_conf` walks the path you specify as its argument and returns the value associated with that path. If there is no value associated with the given path, it returns the name of the next segment in the path. For this reason, configuration information must be defined unambiguously.

For more information, see “Configuring the OPSEC Application” on page 44.

**Prototype**

```
char *opsec_get_conf (OpsecEnv *env, char *path1, char *path2, ...
,..., char *pathn, NULL);
```

**Arguments**

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer to OpsecEnv, as returned by <code>opsec_init</code>.</td>
</tr>
<tr>
<td>path</td>
<td>Each of the path arguments is a non-NULL case-sensitive string (containing no spaces). There may be any number of path arguments, including zero.</td>
</tr>
<tr>
<td>OPSEC_EOL</td>
<td>This indicates the end of the series of string variables.</td>
</tr>
</tbody>
</table>

**Return Values**

A string indicating the value associated with the given path. `NULL` if there is no value associated with the given path.

**Example**

Consider the following configuration file:

```plaintext
my_server port 8080
hi
my_server tower.london.com
abc 1 2 3
```
Managing Entities

The following functions define and destroy OPSEC entities.

opsec_init_entity page 69
opsec_destroy_entity page 74

**opsec_init_entity**

opsec_init_entity defines an entity in the OPSEC environment. Typically, an OPSEC Server defines one entity for each service it provides. An OPSEC Client must define one entity for itself and one for each OPSEC Server with which it will attempt to communicate.

Prototype

OpsecEntity *opsec_init_entity (OpsecEnv *env, OpsecEntityType *entity_type, ...);
### Arguments

**Table 2-10**  
<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer (returned by <code>opsec_init</code>) to the OPSEC environment in which this entity will be created.</td>
</tr>
<tr>
<td>entity_type</td>
<td>One of the following values:</td>
</tr>
<tr>
<td>value</td>
<td>meaning</td>
</tr>
<tr>
<td>AMON_SERVER</td>
<td>AMON Server</td>
</tr>
<tr>
<td>CPMI_CLIENT</td>
<td>CPMI Client</td>
</tr>
<tr>
<td>CPMI_SERVER</td>
<td>CMPI Server</td>
</tr>
<tr>
<td>CVP_SERVER</td>
<td>CVP Server</td>
</tr>
<tr>
<td>ELA_CLIENT</td>
<td>ELA Client</td>
</tr>
<tr>
<td>ELA_SERVER</td>
<td>ELA Server</td>
</tr>
<tr>
<td>LEA_CLIENT</td>
<td>LEA Client</td>
</tr>
<tr>
<td>LEA_SERVER</td>
<td>LEA Server</td>
</tr>
<tr>
<td>UAA_CLIENT</td>
<td>UAA Client</td>
</tr>
<tr>
<td>UAA_SERVER</td>
<td>UAA Server</td>
</tr>
<tr>
<td>UFP_SERVER</td>
<td>UFP Server</td>
</tr>
<tr>
<td>SAM_CLIENT</td>
<td>SAM Client</td>
</tr>
<tr>
<td>SAM_SERVER</td>
<td>SAM Server</td>
</tr>
<tr>
<td>a list of</td>
<td>One of a variable number of attribute-value pairs. Each attribute has zero or more values, with the attribute _OPSEC_EOL (which does not have a value) indicating the end of the list. The attributes are listed in Table 2-11 below.</td>
</tr>
<tr>
<td>attributes and</td>
<td></td>
</tr>
<tr>
<td>their values</td>
<td></td>
</tr>
</tbody>
</table>

The attributes listed in Table 2-11 are common to all OPSEC applications. Other application-specific attributes can be added as well. These are listed in the
documentation for each OPSEC service (CVP, SAM, etc.).

Table 2-11  entity_type values

<table>
<thead>
<tr>
<th>attribute</th>
<th>value type</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPSEC_ENTITY_NAME</td>
<td>char *</td>
<td>The name of the entity (for configuration purposes). It is recommended that there be no white space in this name.</td>
</tr>
<tr>
<td>OPSEC_SESSION_START_HANDLER</td>
<td>handler</td>
<td>The handler that will be called when an OPSEC session begins (see page 127).</td>
</tr>
<tr>
<td>OPSEC_GENERIC_SESSION_START_HANDLER</td>
<td>handler</td>
<td>The handler that will be called when an OPSEC generic session begins.</td>
</tr>
<tr>
<td>OPSEC_SESSION_END_HANDLER</td>
<td>handler</td>
<td>The handler that will be called when an OPSEC session is terminated (see page 130).</td>
</tr>
<tr>
<td>OPSEC_GENERIC_SESSION_END_HANDLER</td>
<td>handler</td>
<td>The handler that will be called when an OPSEC generic session ends.</td>
</tr>
<tr>
<td>OPSEC_SERVER_FAILED_CONN_HANDLER</td>
<td>handler</td>
<td>The handler that will be called if SIC fails while a connection is being established (see page 132). This attribute is valid for only for Server entities.</td>
</tr>
</tbody>
</table>
### OPSEC_SESSION_MULTIPLEX_MODE (available only for client entities)

<table>
<thead>
<tr>
<th>attribute</th>
<th>value type</th>
<th>meaning</th>
</tr>
</thead>
</table>
| OPSEC_SESSION_MULTIPLEX_MODE | int | The maximum number of sessions that may share the same connection (e.g. the same TCP socket). One of:  
  - **MULT_ALL_ON_ONE** if all sessions may share the same connection.  
  - **MULT_ONE_PER_COMM** if every session requires a separate connection.  
  - **MULT_DYNAMIC** if the OPSEC infrastructure may determine the level of multiplexing. This is the default.  
  - An integer representing the number of sessions allowed per connection. |

The values for the following attributes will be overwritten by the values specified using the configuration file or the command line. For more information, see “Configuring the OPSEC Application” on page 44.

- **OPSEC_SERVER_IP**
  - unsigned long
  - The IP address of the Server (network order).

- **OPSEC_ENTITY_SIC_NAME**
  - char *
  - The Server entity's SIC name. For details, “Enabling SIC” on page 43.

- **OPSEC_SERVER_PORT**
  - unsigned long
  - The port on which the Server listens (network order).

- **OPSEC_SERVER_AUTH_PORT**
  - unsigned long
  - The port number for authenticated connections. An OPSEC Server can specify both an authenticated and non-authenticated port. An OPSEC Client can specify only one.
### Table 2-11 entity_type values

<table>
<thead>
<tr>
<th>attribute</th>
<th>value type</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPSEC_SERVER_AUTH_TYPE</td>
<td>unsigned long</td>
<td>Type of authenticated connection as listed in Table 2-12 below — must be specified when OPSEC_SERVER_AUTH_PORT is used.</td>
</tr>
<tr>
<td>OPSEC_SERVER_NO_NAGLE</td>
<td>No value</td>
<td>Disable Nagle's algorithm for the server's connections</td>
</tr>
<tr>
<td>OPSEC_SERVER_CONN_BUF_SIZE</td>
<td>size_t</td>
<td>The maximum buffer size for the entity's connections. See conn_buf_size in table 1-16.</td>
</tr>
</tbody>
</table>

### Table 2-12 connection type values

<table>
<thead>
<tr>
<th>value</th>
<th>NG* based only?</th>
<th>authentication method</th>
<th>encryption method</th>
<th>comp-pressed?</th>
<th>Multithread-Safe?</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPSEC_SSL</td>
<td>*</td>
<td>SSL based</td>
<td>3DES</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>OPSEC_SSLCA_CLEAR</td>
<td>*</td>
<td>certificates</td>
<td>none</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>OPSEC_SSLCA</td>
<td>*</td>
<td></td>
<td>3DES</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>OPSEC_SSLCA_COMP</td>
<td>*</td>
<td></td>
<td></td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>OPSEC_SSLCA_RC4</td>
<td>*</td>
<td></td>
<td>RC4</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>OPSEC_SSLCA_RC4_COMP</td>
<td>*</td>
<td></td>
<td></td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>OPSEC_ASYM_SSLCA</td>
<td>*</td>
<td>asymmetric certificates</td>
<td>3DES</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>OPSEC_ASYM_SSLCA_COMP</td>
<td>*</td>
<td></td>
<td></td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>OPSEC_ASYM_SSLCA_RC4</td>
<td>*</td>
<td></td>
<td>RC4</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>OPSEC_ASYM_SSLCA_RC4_COMP</td>
<td>*</td>
<td></td>
<td></td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>OPSEC_AUTH_SSL_CLEAR</td>
<td>*</td>
<td>SSL based</td>
<td>none</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>OPSEC_SSL_CLEAR</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
</tr>
</tbody>
</table>
Managing Entities

Table 2-12  connection type values

<table>
<thead>
<tr>
<th>value</th>
<th>NG* based only?</th>
<th>authentication method</th>
<th>encryption method</th>
<th>comp-re ssed?</th>
<th>Multithread-Safe?</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPSEC_AUTH_FWN1_AND_SSL</td>
<td>*</td>
<td>Either Check Point proprietary and/or SSL based</td>
<td>none (FWN1) 3DES (SSL)</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>(for Server Applications only)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPSEC_FWN1</td>
<td>*</td>
<td>Check Point proprietary</td>
<td>none</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>OPSEC_AUTH_FWN1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPSEC_NONE</td>
<td>*</td>
<td>none</td>
<td>none</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>OPSEC_LOCAL</td>
<td>*</td>
<td>local</td>
<td></td>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>

*in the second column in table Table 2-12 NG stands for Next Generation.

Return Values
Pointer to OpsecEntity if successful. NULL otherwise.

**opsec_destroy_entity**

opsec_destroy_entity destroys an entity created by opsec_init_entity and frees its memory. All active sessions associated with the entity are terminated at this time.

**Note** -

Destroying the active OpsecEntity (client entity in clients and server entity in servers) will close all open sessions on this entity.

It is forbidden to destroy the OpsecEntity synchronously in a session end handler.

**Prototype**

void opsec_destroy_entity (OpsecEntity *entity);
Managing Sessions

The following functions manage and retrieve information about a given OPSEC session.

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>opsec_new_generic_session</td>
<td>75</td>
</tr>
<tr>
<td>opsec_start_server</td>
<td>76</td>
</tr>
<tr>
<td>opsec_stop_server</td>
<td>76</td>
</tr>
<tr>
<td>opsec_suspend_session_read</td>
<td>77</td>
</tr>
<tr>
<td>opsec_resume_session_read</td>
<td>77</td>
</tr>
<tr>
<td>opsec_end_session</td>
<td>78</td>
</tr>
<tr>
<td>opsec_session_end_reason</td>
<td>78</td>
</tr>
<tr>
<td>opsec_get_own_entity</td>
<td>80</td>
</tr>
<tr>
<td>opsec_get_peer_entity</td>
<td>80</td>
</tr>
<tr>
<td>opsec_get_session_env</td>
<td>80</td>
</tr>
</tbody>
</table>

Recall that the functions that create a service-specific OPSEC session are different for each OPSEC service. The function names are of the form `xxx_new_session`, where `xxx` is an OPSEC service (for example, LEA, SAM etc.). An OPSEC Client initiates a session by calling one of the `xxx_new_session` functions. For more information, see the documentation for the OPSEC Service you are using.

**opsec_new_generic_session**

`opsec_new_generic_session` enables an OPSEC Client to establish a generic (service independent) OPSEC session with an OPSEC Server.
Managing Sessions

Prototype

opsec_new_generic_session (OpsecEntity *client_ent, OpsecEntity *server_ent);

Arguments

Table 2-14  opsec_new_generic_session arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>client_ent</td>
<td>A pointer to the OPSEC Client entity initiating the request.</td>
</tr>
<tr>
<td>entity</td>
<td>A pointer to the OPSEC Server entity with which the session is to be established.</td>
</tr>
</tbody>
</table>

Return Values

Pointer to the new generic session if successful. NULL otherwise.

**opsec_start_server**

opsec_start_server enables an OPSEC Server to begin “listening” for connection attempts by OPSEC Client.

Prototype

int opsec_start_server (OpsecEntity *entity);

Arguments

Table 2-15  opsec_start_server arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>entity</td>
<td>A pointer to the OPSEC Server entity that will start accepting Client requests.</td>
</tr>
</tbody>
</table>

Return Values

0 if successful. -1 otherwise.

**opsec_stop_server**

opsec_stop_server suspends an OPSEC Server. The server will not respond to requests to initiate new sessions, but existing sessions will continue unaffected.
Prototype

```c
int opsec_stop_server (OpsecEntity *entity);
```

Arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>entity</td>
<td>A pointer to the OPSEC Server entity that will stop accepting Client requests.</td>
</tr>
</tbody>
</table>

Return Values

0 if successful. -1 otherwise.

**opsec_suspend_session_read**

`opsec_suspend_session_read` suspends the peer (on the other end of the OPSEC connection) from sending events on all the sessions of a connection.

Prototype

```c
void opsec_suspend_session_read (OpsecSession *session);
```

Arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to an OpsecSession structure.</td>
</tr>
</tbody>
</table>

Return Values

None.

**opsec_resume_session_read**

`opsec_resume_session_read` allows the peer (on the other end of the OPSEC connection) to resume sending events on all sessions of a connection.

Prototype

```c
void opsec_resume_session_read (OpsecSession *session);
```
Managing Sessions

Arguments

Table 2-18  opsec_resume_session_read arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to an OpsecSession structure.</td>
</tr>
</tbody>
</table>

Return Values

None.

**opsec_end_session**

opsec_end_session terminates an OPSEC session and generates an OPSEC_SESSION_END event.

Prototype

void opsec_end_session (OpsecSession *session);

Arguments

Table 2-19  opsec_end_session arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to an OpsecSession structure.</td>
</tr>
</tbody>
</table>

Return Values

None.

**opsec_session_end_reason**

opsec_session_end_reason returns the reason for an OPSEC session's termination.

Prototype

int opsec_session_end_reason (OpsecSession *session);
Arguments

Table 2-20 opsec_session_end_reason arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to an OpsecSession structure.</td>
</tr>
</tbody>
</table>

Return Values

Return values and their meanings are listed in the table below.

Table 2-21 Session Terminated Reasons

<table>
<thead>
<tr>
<th>return value</th>
<th>reason for termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAD_VERSION</td>
<td>Incorrect version at other side of connection.</td>
</tr>
<tr>
<td>COMM_FAILURE</td>
<td>Communication failure.</td>
</tr>
<tr>
<td>COMM_IS_DEAD</td>
<td>No communication.</td>
</tr>
<tr>
<td>END_BY_APPLICATION</td>
<td>The session was ended by the application.</td>
</tr>
<tr>
<td>ENTITY_SESSION_INIT_FAIL</td>
<td>The third-party start handler on other side of connection failed.</td>
</tr>
<tr>
<td>ENTITY_TYPE_SESSION_INIT_FAIL</td>
<td>OPSEC API-specific library code on other side of connection failed when attempting to initialize the session.</td>
</tr>
<tr>
<td>PEER_ENDED</td>
<td>Peer ended the session.</td>
</tr>
<tr>
<td>PEER_SEND_DROP</td>
<td>Peer dropped the connection.</td>
</tr>
<tr>
<td>PEER_SEND_RESET</td>
<td>Peer reset the connection.</td>
</tr>
<tr>
<td>SESSION_NOT_ENDED</td>
<td>The session has not been terminated.</td>
</tr>
<tr>
<td>SESSION_TIMEOUT</td>
<td>The session timeout mechanism has been enabled, the session has timed out, but no session timeout handler has been set. To disable the session timeout mechanism, set the timeout interval to 0 (see “opsec_set_session_timeout” on page 94).</td>
</tr>
<tr>
<td>SIC_FAILURE</td>
<td>The SIC infrastructure was unable to establish the connection—applicable to OPSEC Client applications only.</td>
</tr>
<tr>
<td>UNABLE_TO_ATTACH_COMM</td>
<td>Failure to establish connection.</td>
</tr>
</tbody>
</table>
Managing Sessions

**opsec_get_own_entity**

`opsec_get_own_entity` obtains a pointer to the local (i.e. not peer) entity associated with the session.

**Prototype**

```
OpsecEntity *opsec_get_own_entity (OpsecSession *session);
```

**Arguments**

Table 2-22  `opsec_get_own_entity` arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to an OpsecSession structure.</td>
</tr>
</tbody>
</table>

**Return Values**

Pointer to the local entity if successful. `NULL` otherwise.

**opsec_get_peer_entity**

`opsec_get_peer_entity` obtains a pointer to the peer (i.e. not local) entity associated with the session.

**Prototype**

```
OpsecEntity *opsec_get_peer_entity (OpsecSession *session);
```

**Arguments**

Table 2-23  `opsec_get_peer_entity` arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to an OpsecSession structure.</td>
</tr>
</tbody>
</table>

**Return Values**

Pointer to the peer entity if successful. `NULL` otherwise.

**opsec_get_session_env**

`opsec_get_session_env` obtains a pointer to a session’s `OpsecEnv` structure. You can use this function to access an OPSEC application’s configuration information from within an event handler function.
Prototype

OpsecEnv *opsec_get_session_env (OpsecSession *session);

Arguments

Table 2-24  opsec_get_session_env arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to an OpsecSession structure.</td>
</tr>
</tbody>
</table>

Return Values

Pointer to OpsecEnv structure if successful. NULL otherwise.

Passing Information Between Entities

The following functions manage the OpsecInfo structure, which is used to pass information between OPSEC entities. The data stored in a specific structure are determined by the OPSEC entities that use that structure. Therefore OPSEC Client and Server developers must agree on the data beforehand.

- opsec_info_init (page 81)
- opsec_info_destroy (page 82)
- opsec_info_set (page 82)
- opsec_info_get (page 83)

**opsec_info_init**

opsec_info_init initializes an OpsecInfo structure.

Prototype

OpsecInfo *opsec_info_init ();

Arguments

opsec_info_init does not take any arguments.

Return Values

Pointer to OpsecInfo structure if successful. NULL otherwise.
**opsec_info_destroy**

opsec_info_destroy destroys an OpsecInfo structure and frees its memory.

**Prototype**

```c
void opsec_info_destroy (OpsecInfo *info);
```

**Arguments**

Table 2-25  opsec_info_init arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>info</td>
<td>A pointer to the OpsecInfo structure to be destroyed.</td>
</tr>
</tbody>
</table>

**Return Values**

None.

**opsec_info_set**

opsec_info_set stores data in an OpsecInfo structure.

**Prototype**

```c
int  opsec_info_set (OpsecInfo *info, char *path1, char *path2,
..., char *pathn, char *val, OPSEC_EOL);
```

**Arguments**

Table 2-26  opsec_info_set arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>info</td>
<td>A pointer to OpsecInfo structure.</td>
</tr>
<tr>
<td>path</td>
<td>Each of the path arguments is a non-NULL string (there may be any number of path arguments, including zero).</td>
</tr>
<tr>
<td>val</td>
<td>The value associated with the path made up of all the path values.</td>
</tr>
<tr>
<td>OPSEC_EOL</td>
<td>This indicates the end of the series of string variables.</td>
</tr>
</tbody>
</table>

**Return Values**

0 if successful. Nonzero otherwise.
Example

Suppose you call `opsec_info_set` as follows:

```c
opsec_info_set(info, "elvis", "really", "is", "alive", OPSEC_EOL)
```

Then the path stored in the `OpsecInfo` structure is “elvis really is” and the value of the path is “alive”.

**opsec_info_get**

`opsec_info_get` retrieves data from an `OpsecInfo` structure.

**Prototype**

```c
char *opsec_info_get (OpsecInfo *info, char *path1, char *path2, ...
..., char *pathn, NULL);
```

**Arguments**

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>info</td>
<td>A pointer to <code>OpsecInfo</code> structure.</td>
</tr>
<tr>
<td>path</td>
<td>Each of the path arguments is a non-NULL string (there may be any number of path arguments, including zero).</td>
</tr>
<tr>
<td>NULL</td>
<td>This indicates the end of the series of string variables.</td>
</tr>
</tbody>
</table>

**Return Values**

The value associated with the given path. `NULL` if there is no value associated with the given path.

**Example**

See the example under “opsec_get_conf” on page 67.
Managing the Main Loop, Scheduling, and Socket Events

The following functions initialize the loop at the heart of the OPSEC application, and designate how it should schedule and process events.

- `opsec_mainloop` page 84
- `opsec_schedule` page 85
- `opsec_periodic_schedule` page 86
- `opsec_deschedule` page 86

**opsec_mainloop**

`opsec_mainloop` enables the OPSEC process to start “listening” for OPSEC events. `opsec_mainloop` dispatches all events to the various entities created in the environment.

**Prototype**

```c
int opsec_mainloop (OpsecEnv *env);
```

**Arguments**

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer to the OPSEC environment, as returned by <code>opsec_init</code>.</td>
</tr>
</tbody>
</table>

**Return Values**

Generally, `opsec_mainloop` does not return. However, `opsec mainloop` will return if there are no more events for which to wait. This can happen in the following cases:

- in an OPSEC Client, if there are no active sessions
- in an OPSEC Server, if the Server is not listening for Clients (either because `opsec_start_server` was never called, or because all the started Servers were stopped with `opsec_stop_server` and there are no active sessions)
Managing the Main Loop, Scheduling, and Socket Events

0 if successful. Nonzero otherwise.

**Note** - An instance of an OPSEC environment can not be used to run an OPSEC main loop more than once in its lifetime. If an OPSEC main loop needs to be rerun, the OPSEC environment should be deleted and re-created.

**opsec_schedule**

opsec_schedule schedules a function to be executed the next time opsec_mainloop takes control, but not before a specified interval has elapsed.

**Prototype**

```c
void opsec_schedule (OpsecEnv *env, long interval, 
void(*fn)(void*), void *arg);
```

**Arguments**

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer to the environment passed to opsec_mainloop.</td>
</tr>
<tr>
<td>interval</td>
<td>The delay before the scheduled function will be invoked, in milliseconds. It is recommended that you specify this argument as long.</td>
</tr>
<tr>
<td>fn</td>
<td>A pointer to the scheduled function (which takes a void pointer as its only argument).</td>
</tr>
<tr>
<td>arg</td>
<td>The argument to the scheduled function.</td>
</tr>
</tbody>
</table>

**Return Values**

None.

**Notes**

The scheduled function is uniquely identified by its fn and arg arguments. If you call opsec_schedule or opsec_periodic_schedule a second time with the same fn and arg arguments, then the second call overrides the first.

For example, suppose you call opsec_schedule as follows:

```c
opsec_schedule(env, 10L, printf, (void *) pFormat);
opsec_schedule(env, 1000L, printf, (void *) pFormat);
```
then printf will be scheduled only once, with an interval of 1000 milliseconds.

On the other hand, suppose you call \texttt{opsec\_schedule} as follows:

\begin{verbatim}
  opsec_schedule(env, 10L, printf, (void *) pFormat1);
  opsec_schedule(env, 10L, printf, (void *) pFormat2);
\end{verbatim}

In this case, printf will be scheduled twice.

\textbf{\texttt{opsec\_periodic\_schedule}}

\texttt{opsec\_periodic\_schedule} schedules a function to be called repeatedly at the specified interval once \texttt{opsec\_mainloop} takes control.

\textbf{Prototype}

\begin{verbatim}
void opsec_periodic_schedule (OpsecEnv *env, long interval, 
    void(*fn)(void*), void *arg);
\end{verbatim}

\textbf{Arguments}

\begin{table}[h]
\begin{tabular}{|l|l|}
\hline
\textbf{argument} & \textbf{meaning} \\
\hline
env & A pointer to the environment passed to \texttt{opsec\_mainloop}. \\
interval & The interval, in milliseconds, between function calls. It is recommended that you explicitly specify this argument as type \texttt{long}. \\
fn & A pointer to the scheduled function (which takes a void pointer as its only argument). \\
arg & The argument to the scheduled function. \\
\hline
\end{tabular}
\end{table}

\textbf{Return Values}

None.

\textbf{Notes}

See “Notes” on page 85, under \texttt{opsec\_schedule}.

\textbf{\texttt{opsec\_deschedule}}

\texttt{opsec\_deschedule} deschedules a function execution previously scheduled by \texttt{opsec\_schedule} or \texttt{opsec\_periodic\_schedule}. 
Prototype

```c
void opsec_deschedule (OpsecEnv *env, void(*fn)(void*), void *arg);
```

Arguments

<table>
<thead>
<tr>
<th>Table 2-31</th>
<th>opsec_deschedule arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>argument</td>
<td>meaning</td>
</tr>
<tr>
<td>env</td>
<td>A pointer to the environment passed to opsec_mainloop.</td>
</tr>
<tr>
<td>fn</td>
<td>A pointer to the scheduled function (which takes a void pointer as its only argument).</td>
</tr>
<tr>
<td>arg</td>
<td>The argument to the scheduled function.</td>
</tr>
</tbody>
</table>

Return Values

None.

Notes

The scheduled function is uniquely identified by the fn and arg arguments. If no such function is currently scheduled (perhaps because it has already been executed) then opsec_deschedule does nothing.

opsec_set_socket_event

opsec_set_socket_event defines the handler function to be called when the specified event occurs. This enables opsec_mainloop to handle events that are not part of the OPSEC environment.

The handler function is uniquely identified by its event and sock arguments. If a handler function is already defined for these arguments then the new definition replaces the previous one.

Prototype

```c
void opsec_set_socket_event (OpsecEnv *env, int event, int sock, int(*fn)(int, void *ptr), void *ptr);
```
Managing the Main Loop, Scheduling, and Socket Events

Arguments

Table 2-32 opsec_set_socket_event arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer to the environment passed to opsec_mainloop.</td>
</tr>
<tr>
<td>event</td>
<td>One of the following values:</td>
</tr>
<tr>
<td></td>
<td>value</td>
</tr>
<tr>
<td></td>
<td>OPSEC_SK_INPUT</td>
</tr>
<tr>
<td></td>
<td>OPSEC_SK_OUTPUT</td>
</tr>
<tr>
<td></td>
<td>OPSEC_SK_EXCEPTIONAL</td>
</tr>
<tr>
<td>sock</td>
<td>The socket number.</td>
</tr>
<tr>
<td>fn</td>
<td>A pointer to the handler function that will be called in response to the event. The handler function takes two arguments: sock and ptr (the socket number and a pointer to the event’s data).</td>
</tr>
<tr>
<td>ptr</td>
<td>A pointer to the event data.</td>
</tr>
</tbody>
</table>

Return Values

None.

**opsec_del_socket_event**

opsec_del_socket_event deletes a definition set by opsec_set_socket_event. The handler function is uniquely identified by its event and sock arguments. If no such handler function is currently defined then opsec_del_socket_event does nothing.

Prototype

void opsec_del_socket_event (OpsecEnv *env, int event, int sock);
Arguments

Table 2-33 opsec_del_socket_event arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer to the environment passed to opsec_mainloop.</td>
</tr>
<tr>
<td>event</td>
<td>One of the following values:</td>
</tr>
<tr>
<td>value</td>
<td>meaning</td>
</tr>
<tr>
<td></td>
<td>OPSEC_SK_INPUT</td>
</tr>
<tr>
<td></td>
<td>An incoming event on the given socket.</td>
</tr>
<tr>
<td></td>
<td>OPSEC_SK_OUTPUT</td>
</tr>
<tr>
<td></td>
<td>An outgoing event on the given socket.</td>
</tr>
<tr>
<td></td>
<td>OPSEC_SK_EXCEPTIONAL</td>
</tr>
<tr>
<td></td>
<td>An out-of-band event on the given socket.</td>
</tr>
<tr>
<td>sock</td>
<td>The socket number.</td>
</tr>
</tbody>
</table>

Return Values

None.

OPSEC Utilities

The following functions enable an OPSEC entity to measure peer response time, generate TCP traffic to keep a session alive, set session timeout, and retrieve information about an OPSEC session. These functions are supported by both generic and service-specific sessions.

opsec_start_keep_alive page 90
opsec_stop_keep_alive page 90
opsec_free page 91
opsec_ping_peer page 92
opsec_set_session_timeout page 94
opsec_set_session_timeout_handler page 95
opsec_get_sdk_version page 96
opsec_get_peer_sdk_version page 96
opsec_get_local_address page 97
opsec_get_peer_address page 98
**opsec_start_keep_alive**

`opsec_start_keep_alive` starts the OPSEC keep-alive service. This service generates traffic on a TCP connection whenever there is no other traffic for the designated interval, thereby enabling an OPSEC entity to avoid TCP level timeouts.

**Prototype**

```
int opsec_start_keep_alive (OpsecSession *session, int interval);
```

**Arguments**

**Table 2-34  `opsec_start_keep_alive` arguments**

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to the OPSEC session to be “kept alive.”</td>
</tr>
<tr>
<td>interval</td>
<td>The maximum amount of time (in milliseconds) that a session may be silent before new traffic is generated.</td>
</tr>
</tbody>
</table>

**Return Values**

0 if successful. Nonzero otherwise.

**opsec_stop_keep_alive**

`opsec_stop_keep_alive` stops the OPSEC keep-alive service.

**Prototype**

```
int opsec_stop_keep_alive (OpsecSession *session);
```

**Arguments**

**Table 2-35  `opsec_stop_keep_alive` arguments**

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to the session that is being “kept alive.”</td>
</tr>
</tbody>
</table>

**Return Values**

0 if successful. Nonzero otherwise.
**opsec_free**

opsec_free frees memory allocated by the OPSEC library; the memory does not have a specific destruction API.

**Prototype**

```c
void opsec_free (void *p);
```

**Arguments**

**Table 2-36  opsec_free arguments**

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>A pointer to the memory to be freed.</td>
</tr>
</tbody>
</table>

**Return Values**

None.

**opsec_DllMain**

opsec_DllMain performs initialization and clean-up, and should be used in the following cases:

- If the OPSEC application is multithreaded, and it is an executable that statically links to static opsec libraries. In this case, each thread, before exit, should call opsec_DllMain with the following arguments:
  ```c
  opsec_DllMain (NULL, DLL_THREAD_DETACH, NULL);
  ```
- If the OPSEC application is a dynamically linked library that wraps up static opsec libraries. In this case, whenever DllMain is called by the operating system, opsec_DllMain should be called with the same arguments. For further information on DllMain, see the Microsoft documentation.

**Prototype**

```c
BOOL WINAPI opsec_DllMain (HINSTANCE hinstDLL,
DWORD fdwReason,
```
LPVOID lpReserved);

Note - This function is defined only for WIN32 static libraries.

Arguments
See the Microsoft documentation.

Return Values
See the Microsoft documentation.

opsec_ping_peer

opsec_ping_peer sends small fixed data packets to the OPSEC peer, and measures
the time elapsed until the peer responds.

Prototype
int opsec_ping_peer (OpsecSession *session, int timeout,
void (*handler)(OpsecSession *sess, u_int mask,
OpsecInfo *info, long rtime, int status,
void *opaque)
void *opaque);
Arguments

Table 2-37  opsec_ping_peer arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to the session with the OPSEC entity to be pinged.</td>
</tr>
<tr>
<td>timeout</td>
<td>The time (in milliseconds) that the pinging OPSEC entity is willing to wait for the callback message. 0 if the pinging entity is not expecting a callback on timeout expiration.</td>
</tr>
<tr>
<td>handler</td>
<td>A pointer to the handler function that will be called when a reply arrives from the pinged entity. This function must have the following arguments:</td>
</tr>
<tr>
<td>argument</td>
<td>meaning</td>
</tr>
<tr>
<td>sess</td>
<td>A pointer to the session.</td>
</tr>
<tr>
<td>mask</td>
<td>Reserved for future use. A bitmask to be used with info.</td>
</tr>
<tr>
<td>info</td>
<td>Reserved for future use. An OpsecInfo structure containing additional parameters.</td>
</tr>
<tr>
<td>rtime</td>
<td>The round trip time in seconds.</td>
</tr>
</tbody>
</table>
| status   | One of:  
- PING_PEER_STAT_OK - received reply to ping request  
- PING_PEER_STAT_SEND_ERR - unable to send ping request to server  
- PING_PEER_STAT_TIMEOUT - no reply received within specified time. |
| opaque   | User-supplied data to be passed to the callback function. |

Return Values

0 if successful. Nonzero otherwise.
Notes

A given session will not run more than one ping request at a time. All subsequent ping requests will be rejected until the original request receives a response or times out. `opsec_ping_peer` will return 0 (success) even if the request is rejected.

**opsec_set_session_timeout**

`opsec_set_session_timeout` sets the time limit beyond which a session will expire if no new data is received. This timeout is only valid for the current session.

When the specified amount of time has elapsed, one of the following events takes place:

- If there is no registered timeout handler, the session is closed, and `SESSION_TIMEOUT` is given as the end reason.
- If there is a registered handler, it is called. This handler then determines whether the session is to be closed or not.

For more information about the timeout handler, see “opsec_set_session_timeout_handler” on page 95.

The timer is reset by any of the following events:

- The OPSEC entity receives a service-specific message
- The OPSEC entity is pinged by its peer.
- The timer has expired, but the session is not closed.

Note that the keep-alive service does not reset the timer.

Prototype

```c
int opsec_set_session_timeout (OpsecSession *session, int timeout);
```
Arguments

Table 2-38  opsec_set_session_timeout arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to the session for which the timeout is to be set.</td>
</tr>
<tr>
<td>timeout</td>
<td>The maximum amount of time, in milliseconds, that a session may be silent before it times out. If 0, this timeout mechanism is disabled and the timeout handler is not called.</td>
</tr>
</tbody>
</table>

Return Values

0 if successful. Nonzero otherwise.

**opsec_set_session_timeout_handler**

opsec_set_session_timeout_handler registers the handler to be called with the OPSEC session times out.

Prototype

```c
int opsec_set_session_timeout_handler (OpsecSession *session, int(*)(OpsecSession *));
```

Arguments

Table 2-39  opsec_set_session_timeout_handler arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to the session for which the handler is to be registered.</td>
</tr>
</tbody>
</table>
| handler  | A pointer to the handler function. This function must receive a pointer to the expired session as its argument, and must return one of the following values:
|          | • OPSEC_SESSION_END if the session is to be closed
|          | • OPSEC_SESSION_OK if the session is to continue |

Return Values

0 if successful. 1 otherwise.
**opsec_get_sdk_version**

`opsec_get_sdk_version` obtains the SDK version and patch number used on this end of the connection.

**Prototype**

```c
void opsec_get_sdk_version (int *sdk_ver, int *patch_num, int *build_num,
char **ver_desc, char **full_desc);
```

**Arguments**

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>sdk_ver</td>
<td>The version number, calculated as (major version * 1000) + (minor version (usually a Service Pack))</td>
</tr>
<tr>
<td>patch_num</td>
<td>The patch number.</td>
</tr>
<tr>
<td>build_num</td>
<td>The build number.</td>
</tr>
<tr>
<td>ver_desc</td>
<td>Version description (for example, “OPSEC SDK 4.1”).</td>
</tr>
<tr>
<td>full_desc</td>
<td>Full description (For example, “Opsec SDK 4.1 patch = 1010 build = 11 Bug fix release”).</td>
</tr>
</tbody>
</table>

**Return Values**

None.

**opsec_get_peer_sdk_version**

`opsec_get_peer_sdk_version` obtains the SDK version and patch number of the OPSEC peer (on the other end of the OPSEC connection).

**Prototype**

```c
int opsec_get_peer_sdk_version (OpsecSession *session,
void (*f)(OpsecSession *,int,int,char *,char *));
```
Arguments

Table 2-41  opsec_get_peer_sdk_version arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to an OpsecSession structure.</td>
</tr>
<tr>
<td>f</td>
<td>A callback function that will be called with peer version reply. The parameters of f are: SDK version, patch number, build number, version description and full description.</td>
</tr>
</tbody>
</table>

Return Values

-1 if failure. Otherwise successful.

**opsec_get_local_address**

opsec_get_local_address retrieves an OPSEC session’s communication type, local IP address, and local port number.

Prototype

```c
int opsec_get_local_address (OpsecSession *session, int *comm_type, u_int *ip, u_short *port)
```

Arguments

Table 2-42  opsec_get_local_address arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to an OpsecSession structure.</td>
</tr>
<tr>
<td>comm_type</td>
<td>One of the following values:</td>
</tr>
<tr>
<td>value</td>
<td>meaning</td>
</tr>
<tr>
<td>OPSEC_TCP_COMM</td>
<td>non-authenticated communication</td>
</tr>
<tr>
<td>ip</td>
<td>The local IP address for the connection in network order.</td>
</tr>
<tr>
<td>port</td>
<td>The local port for the connection in network order.</td>
</tr>
</tbody>
</table>

Return Values

0 if successful. Nonzero otherwise.
**opsec_get_peer_address**

`opsec_get_peer_address` retrieves information about the OPSEC session peer (on the other end of the OPSEC connection), including the type of communication, the peer’s IP address, and the port number he uses for communication. This function should be called from an event handler (but not from the `OPSEC_SESSION_START` event handler).

**Prototype**

```c
int opsec_get_peer_address (OpsecSession *session, int *comm_type, u_int *ip, u_short *port)
```

**Arguments**

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to an <code>OpsecSession</code> structure.</td>
</tr>
<tr>
<td>comm_type</td>
<td>One of the following values:</td>
</tr>
<tr>
<td>value</td>
<td>meaning</td>
</tr>
<tr>
<td>OPSEC_TCP_COMM</td>
<td>non-authenticated communication</td>
</tr>
<tr>
<td>ip</td>
<td>The peer’s IP address in network order.</td>
</tr>
<tr>
<td>port</td>
<td>The peer port for the connection in network order.</td>
</tr>
</tbody>
</table>

**Return Values**

0 if successful. Nonzero otherwise.
SIC Utilities

The following functions enable an OPSEC entity to retrieve its own SIC name, its peer SIC name, and the SIC method.

opsec_sic_get_sic_service

opsec_sic_get_sic_service returns the SIC service used by the given OPSEC session.

Prototype

char *

opsec_sic_get_sic_service(OpsecSession *session, short *service_num);

Arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to an OPSEC session.</td>
</tr>
<tr>
<td>service_num</td>
<td>An optional pointer for a output parameter through which the service number (which corresponds to the service name) returned by the function is returned. A NULL pointer can be passed if the output parameter is of no interest.</td>
</tr>
</tbody>
</table>

Return Values

The service name if successful NULL otherwise. See Table 1-18 and Table 1-19 for a list of service names. The service_num output parameter will not be set on failure (if the function return value is NULL).
**opsec_add_sic_rule**

`opsec_add_sic_rule` adds a SIC rule to the SIC policy loaded into the application address space (not to the SIC policy file that may be in use by the application). The SIC rule is always added at the top of the already existing SIC rules (that are traversed from top to bottom when a search for a matching SIC rule is done).

**Prototype**

```c
int opsec_add_sic_rule(OpsecEnv *env, int direction, char *apply_to, char *peer, char *dst_port, char *svc_name, char *method_name, void **rule );
```
Arguments

Table 2-45  opsec_add_sic_rule arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer to an OPSEC environment</td>
</tr>
<tr>
<td>direction</td>
<td>An integer defining the SIC rule group to be added with the new rule. 0 for Inbound, 1 for Outbound.</td>
</tr>
<tr>
<td>apply_to</td>
<td>A string defining the SIC name of the application for which this rule should apply (normally set to ‘ANY’ or ‘ME’).</td>
</tr>
<tr>
<td>peer</td>
<td>A string defining the SIC name of the peer.</td>
</tr>
<tr>
<td>dest_port</td>
<td>A string representation of the port number to which the connection is being made.</td>
</tr>
<tr>
<td>svc_name</td>
<td>The name of the SIC service used. See Table 1-18 and Table 1-19 for a list of service names.</td>
</tr>
<tr>
<td>method_name</td>
<td>The name of the SIC method used. See Table 1-18 and Table 1-19 for a list of SIC Methods.</td>
</tr>
<tr>
<td>rule</td>
<td>An optional pointer for an output parameter through which the handle of the rule will be returned.</td>
</tr>
</tbody>
</table>

Return Values

0 on success. < 0 on failure.

opsec_delete_sic_rule

opsec_delete_sic_rule deletes a rule from the SIC policy loaded into the application address space (not to the SIC policy file that may be in use by the application).

Prototype

```c
int opsec_delete_sic_rule(OpsecEnv *env, void *rule);
```
Arguments

Table 2-46  opsec_delete_sic_rule arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer to an OPSEC environment</td>
</tr>
<tr>
<td>rule</td>
<td>A handle to an added SIC rule obtained from the “opsec_add_sic_rule” API</td>
</tr>
</tbody>
</table>

Return Values

0 on success, -1 on failure.

**opsec_get_my_sic_name**

opsec_get_my_sic_name returns the local SIC name set for the OPSEC environment.

Prototype

char * opsec_get_my_sic_name (OpsecEnv *env);

Arguments

Table 2-47  opsec_get_my_sic_name arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer to the OPSEC environment.</td>
</tr>
</tbody>
</table>

Return Values

The local entity’s SIC name if successful. NULL otherwise.

In multi-sic applications (even if its an application with 1 identity), opsec_get_my_sic_name returns NULL.

**opsec_sic_get_peer_sic_name**

opsec_sic_get_peer_sic_name returns the SIC name of the peer if the designated session is an SIC session.

Prototype

char * opsec_sic_get_peer_sic_name (OpsecSession *session);
Arguments

Table 2-48  \texttt{opsec\_sic\_get\_peer\_sic\_name} arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to an \texttt{OpsecSession} structure.</td>
</tr>
</tbody>
</table>

Return Values

The peer’s SIC name if successful. \texttt{NULL} otherwise.

\texttt{NULL} is also returned if the peer’s SIC name is not known (e.g. the session has not been fully established).

\texttt{opsec\_sic\_get\_sic\_method}

\texttt{opsec\_sic\_get\_sic\_method} returns the SIC method chosen if the designated session is an SIC session.

Prototype

\texttt{char \* opsec\_sic\_get\_sic\_method (OpsecSession \*session);}

Arguments

Table 2-49  \texttt{opsec\_sic\_get\_sic\_method} arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to an \texttt{OpsecSession} structure.</td>
</tr>
</tbody>
</table>

Return Values

The SIC method if successful. \texttt{NULL} otherwise.

\texttt{NULL} is also returned if the SIC negotiations have not been completed, and a method has not yet been chosen.

\texttt{opsec\_sic\_get\_peer\_cert\_hash}

\texttt{opsec\_sic\_get\_peer\_cert\_hash} retrieves the certificate hash of the OPSEC session peer (on the other end of the OPSEC connection).
Prototype

```c
int opsec_sic_get_peer_cert_hash (OpsecSession *session, u_char *hash,
                                 u_int *hash_len, char *cert_str, int cert_str_len)
```

Arguments

Table 2-50  opsec_sic_get_peer_cert_hash arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to an OpsecSession structure.</td>
</tr>
<tr>
<td>hash</td>
<td>A pointer to the buffer in which the hash is to be returned.</td>
</tr>
<tr>
<td>hash_len</td>
<td>A pointer to the integer containing the length of the hash buffer. This parameter will be set to the correct length regardless of whether the hash buffer is sufficiently long.</td>
</tr>
<tr>
<td>cert_str</td>
<td>A pointer to the buffer in which the certificate string (i.e. SKEY words) is to be returned.</td>
</tr>
<tr>
<td>cert_str_len</td>
<td>The length of the certificate string buffer.</td>
</tr>
</tbody>
</table>

Return Values

0 if successful. -1 if any of the following is true:

- Any of the pointers are invalid.
- The authentication type set for the specified session does not require certificates.
- The hash buffer is too short.
- There is a problem obtaining the hash from the certificate.
- The certificate string buffer is too short.
Macros

This section describes each of the common OPSEC API macros in detail. The type definitions are located in the `opsec.h` file.

**SESSION_OPAQUE(session)**

This macro can be used for storing application specific data (`void`) for a session.

A typical use for this macro would be to store a pointer to a structure. In the StartHandler (see “Event Handler for the OPSEC_SESSION_START event” on page 127), you can allocate the memory for the data as follows:

```c
SESSION_OPAQUE(session) = malloc (sizeof((MyPrivateData)...)
```

You can then use the pointer freely in your application. For example:

```c
((MyPrivateData *)SESSION_OPAQUE(session))->counter = 0;
.saved_counter = ((MyPrivateData *)SESSION_OPAQUE(session))->counter;
```

When finished, free the allocated memory in the EndHandler as follows:

```c
if (SESSION_OPAQUE(session))
    free ((SESSION_OPAQUE(session));
```
Error Handling Functions

This section describes error handling functions. The function prototypes are defined in `opsec_error.h`.

```
opsec_errno_str page 106
opsec_get_sic_error page 106
```

**opsec_errno_str**

`opsec_errno_str` converts an OPSEC error number to a string. The function prototype is defined in `opsec_error.h`.

**Prototype**

```
char *opsec_errno_str (int err);
```

**Arguments**

Table 2-51  
opsec_errno_str arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>err</td>
<td>An OPSEC error code. Normally this will be the global variable <code>opsec_errno</code>. This variable is defined only if the file <code>opsec_error.h</code> is included.</td>
</tr>
</tbody>
</table>

**Return Values**

Pointer to a static string. Do not free this pointer.

**opsec_get_sic_error**

`opsec_get_sic_error` retrieves SIC error information for a specific session. The function prototype is defined in `opsec.h`.

```
Note - opsec_get_sic_error is only meaningful if the session was established using the SIC framework. If the session was established using the port keyword rather than auth_port, then the SIC framework is not used and opsec_get_sic_error will not return valid SIC error information.
```

**Prototype**

```
int opsec_get_sic_error (OpsecSession *session, int *sic_errno, char **sic_errmsg);
```
Arguments

Table 2-52  \texttt{opsec\_get\_sic\_error} arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to an OpsecSession structure.</td>
</tr>
<tr>
<td>sic_errno</td>
<td>A pointer to be set to the SIC error code.</td>
</tr>
<tr>
<td>sic_errmsg</td>
<td>A pointer to be set to the SIC error message.</td>
</tr>
</tbody>
</table>

Return Values

Zero if successful. Nonzero if session is not valid or if SIC is not in use for this session.
Events API Functions

This section describes each of the Events API functions in detail. The function prototypes are defined in opsec_event.h.

Managing Events

The following functions create event classes, and manage event handlers. To define an event handler, see “General Event Handlers” on page 133.

- opsec_new_event_id
- opsec_set_event_handler
- opsec_del_event_handler
- opsec_suspend_event_handler
- opsec_resume_event_handler

An event handler is uniquely defined by the following:

- an event class identifier
- the event handler function
- the data passed to the function when the event handler is defined by opsec_set_event_handler

In other words, the arguments (event_num, handler, and set_data) to the event handler functions described in this section uniquely identify an event handler.

Note - OPSEC events use messaging queues to send the OPSEC events from one thread to the other. The queues may become congested. The OPSEC event functions make a distinction between a general failure and a failure related to message queue congestion.

opsec_new_event_id

opsec_new_event_id creates an event class.

Prototype

long opsec_new_event_id();
Managing Events

Arguments

**opsec_new_event_id** takes no arguments.

Return Values

A unique event identifier that can be used to specify this event class in other event functions.

**opsec_set_event_handler**

**opsec_set_event_handler** defines a handler for the specified event class. Any number of handlers can be defined for an event class, and they will be called one after the other in the sequence in which they were defined by successive calls to **opsec_set_event_handler**.

Prototype

```c
int opsec_set_event_handler (OpsecEnv *env, long event_num, OpsecEventHandler handler, void *set_data);
```

Arguments

Table 2-53  **opsec_set_event_handler** arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer to the environment passed to <strong>opsec_mainloop</strong>.</td>
</tr>
<tr>
<td>event_num</td>
<td>The event identifier (returned by <strong>opsec_new_event_id</strong>).</td>
</tr>
<tr>
<td>handler</td>
<td>A pointer to the event handler function (see “General Event Handlers” on page 133).</td>
</tr>
<tr>
<td>set_data</td>
<td>A pointer to the data passed to handler.</td>
</tr>
</tbody>
</table>

Return Values

0 for success, -1 for failure, -2 for failure because of congestion.

**opsec_del_event_handler**

**opsec_del_event_handler** deletes a handler for the specified event class.
Prototype

```c
int opsec_del_event_handler (OpsecEnv *env, long event_num, OpsecEventHandler handler, void *set_data);
```

Arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer to the environment passed to opsec_mainloop.</td>
</tr>
<tr>
<td>event_num</td>
<td>The event identifier (returned by opsec_new_event_id).</td>
</tr>
<tr>
<td>handler</td>
<td>A pointer to the event handler function (see “General Event Handlers” on page 133).</td>
</tr>
<tr>
<td>set_data</td>
<td>A pointer to the data passed to handler.</td>
</tr>
</tbody>
</table>

Return Values

0 for success, -1 for failure, -2 for failure because of congestion.

**opsec_suspend_event_handler**

*opsec_suspend_event_handler* suspends the handling of events of the given class by the specified handler. Subsequent events are stored and will be processed only after *opsec_resume_event_handler* is called.

Prototype

```c
int opsec_suspend_event_handler (OpsecEnv *env, long event_num, OpsecEventHandler handler, void *set_data);
```
Managing Events

Chapter 2  OPSEC API (Application Programming Interface)  111

Arguments

Table 2-55  opsec_suspend_event_handler arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer to the environment passed to opsec_mainloop.</td>
</tr>
<tr>
<td>event_num</td>
<td>The event identifier (returned by opsec_new_event_id).</td>
</tr>
<tr>
<td>handler</td>
<td>A pointer to the event handler function (see “General Event Handlers” on page 133).</td>
</tr>
<tr>
<td>set_data</td>
<td>A pointer to the data passed to handler.</td>
</tr>
</tbody>
</table>

Return Values

0 for success, -1 for failure.

**opsec_resume_event_handler**

opsec_resume_event_handler undoes the effects of opsec_suspend_event_handler. Any pending events (that occurred while opsec_suspend_event_handler was in effect) are then processed by the handler in the sequence in which they occurred.

Prototype

```c
int opsec_resume_event_handler (OpsecEnv *env, long event_num, OpsecEventHandler handler, void *set_data);
```

Arguments

Table 2-56  opsec_resume_event arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer to the environment passed to opsec_mainloop.</td>
</tr>
<tr>
<td>event_num</td>
<td>The event identifier (returned by opsec_new_event_id).</td>
</tr>
<tr>
<td>handler</td>
<td>A pointer to the event handler function (see “General Event Handlers” on page 133).</td>
</tr>
<tr>
<td>set_data</td>
<td>A pointer to the data passed to handler.</td>
</tr>
</tbody>
</table>

Return Values

0 for success, -1 for failure, -2 for failure because of congestion.
Raising and Unraising Event Instances

The following functions raise (trigger) and unraise events.

- `opsec_raise_event`  
- `opsec_raise_persistent_event`  
- `opsec_unraise_event`  
- `opsec_unraise_event`

See “Events API Overview” on page 34 for an explanation of the difference between a regular event and a persistent event.

An event instance is uniquely defined by the following:

- the OPSEC process’s environment
- a unique event identifier
- the data set when the event is raised

The arguments (env, event_num, and raise_data) to each of the functions described in this section uniquely identify an event instance.

**opsec_raise_event**

`opsec_raise_event` raises a regular event instance. If multiple events are raised using the same event_num and raise_data—that is, if the events are identical—then the handler is called only once.

If multiple event handlers are defined for a given event class, they will be called one after the other in the sequence in which they were defined by successive calls to `opsec_set_event_handler`.

**Prototype**

```c
int opsec_raise_event (OpsecEnv *env, long event_num, void *raise_data);
```
Arguments

Table 2-57   opsec_raise_event arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer to the environment passed to opsec_mainloop.</td>
</tr>
<tr>
<td>event_num</td>
<td>The event identifier, as returned by opsec_new_event_id.</td>
</tr>
<tr>
<td>raise_data</td>
<td>A pointer to the data passed to the event handler (see “General Event Handlers” on page 133).</td>
</tr>
</tbody>
</table>

Return Values

0 if successful. -1 if there is no handler registered for this event or -2 for failure because of congestion.

**opsec_raise_persistent_event**

opsec_raise_persistent_event raises a persistent event instance.

If multiple event handlers are defined for a given event class, they will be called one after the other in the sequence in which they were defined by successive calls to opsec_set_event_handler.

Prototype

```c
int opsec_raise_persistent_event (OpsecEnv *env, long event_num, void *raise_data);
```

Arguments

Table 2-58   opsec_raise_persistent_event arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer to the environment passed to opsec_mainloop.</td>
</tr>
<tr>
<td>event_num</td>
<td>The event identifier, as returned by opsec_new_event_id.</td>
</tr>
<tr>
<td>raise_data</td>
<td>A pointer to the data passed to the event handler (see “General Event Handlers” on page 133).</td>
</tr>
</tbody>
</table>

Return Values

0 if successful. -1 otherwise or -2 for failure because of congestion.
Notes

To enable your API thread to exit its `opsec_mainloop` function, you must call `opsec_unraise_event()` to de-register each persistent event.

**opsec_unraise_event**

`opsec_unraise_event` unraises an event instance, that is, “undoes” an `opsec_raise_event` or `opsec_raise_persistent_event`.

**Prototype**

```c
int opsec_unraise_event (OpsecEnv *env, long event_num, void *raise_data);
```

**Arguments**

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer to the environment passed to <code>opsec_mainloop</code>.</td>
</tr>
<tr>
<td>event_num</td>
<td>The event identifier, as returned by <code>opsec_new_event_id</code>.</td>
</tr>
<tr>
<td>raise_data</td>
<td>A pointer to the data passed to the event handler</td>
</tr>
<tr>
<td></td>
<td>(see “General Event Handlers” on page 133).</td>
</tr>
</tbody>
</table>

**Return Values**

0 for success, -1 for failure, -2 for failure because of congestion.

**opsec_israised_event**

`opsec_israised_event` queries the status of an event instance.

**Prototype**

```c
int opsec_israised_event (OpsecEnv *env, long event_num, void *raise_data);
```
Arguments

Table 2-60  opsec_israised_event arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer to the environment passed to opsec_mainloop.</td>
</tr>
<tr>
<td>event_num</td>
<td>The event identifier, as returned by opsec_new_event_id.</td>
</tr>
<tr>
<td>raise_data</td>
<td>A pointer to the data passed to the event handler (see “General Event Handlers” on page 133).</td>
</tr>
</tbody>
</table>

Return Values

Nonzero if the event is raised. 0 if the event is not raised, meaning that it has either already been processed or unraised.
OPSEC Unique IDs

This section describes each of Unique ID API (UID) functions in detail. The function prototypes are defined in `opsec_uuid.h`.

The following functions create, destroy and handle Unique IDs.

- `opsec_uuid_create` creates a new Unique ID. A new UID is created as an "unspecified" UID.
- `opsec_uuid_destroy` destroys a previously created Unique ID.
- `opsec_uuid_duplicate`
- `opsec_uuid_set_unspecified`
- `opsec_uuid_equal`
- `opsec_uuid_to_string`
- `opsec_uuid_from_string`

**opsec_uuid_create**

`opsec_uuid_create` creates a new Unique ID. A new UID is created as an "unspecified" UID.

**Prototype**

```c
opsec_uuid *opsec_uuid_create();
```

**Arguments**

None.

**Return Values**

A pointer to an `opsec_uuid` upon success, NULL upon failure.

**opsec_uuid_destroy**

`opsec_uuid_destroy` destroys a previously created Unique ID.

**Prototype**

```c
void opsec_uuid_destroy (opsec_uuid *uuid);
```
Arguments

Table 2-61  opsec_uuid_destroy arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>A pointer to a previously created Unique ID.</td>
</tr>
</tbody>
</table>

Return Values

None.

`opsec_uuid_duplicate`

`opsec_uuid_duplicate` duplicates a Unique ID.

Prototype

```c
opsec_uuid *opsec_uuid_duplicate (opsec_uuid *uuid);
```

Arguments

Table 2-62  opsec_uuid_duplicate arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>A pointer to a previously created Unique ID.</td>
</tr>
</tbody>
</table>

Return Values

A pointer to an `opsec_uuid` upon success, NULL upon failure.

`opsec_uuid_set_unspecified`

`opsec_uuid_set_unspecified` sets a Unique ID to “unspecified”.

Prototype

```c
int opsec_uuid_set_unspecified (opsec_uuid *uuid);
```
Arguments

Table 2-63  opsec_uuid_set_unspecified arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>A pointer to a previously created Unique ID.</td>
</tr>
</tbody>
</table>

Return Values

EO_OK upon success, EO_ERROR upon failure.

**opsec_uuid_equal**

opsec_uuid_equal compares the first Unique ID to the second Unique ID.

Prototype

```c
int opsec_uuid_equal(opsec_uuid *uuid1, opsec_uuid *uuid2);
```

Arguments

Table 2-64  opsec_uuid_equal arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid1</td>
<td>Pointer to the first Unique ID to be compared.</td>
</tr>
<tr>
<td>uuid2</td>
<td>Pointer to the second Unique ID to be compared.</td>
</tr>
</tbody>
</table>

Return Values

1 if equal, 0 is unequal or -1 upon error.

**opsec_uuid_to_string**

opsec_uuid_to_string converts a unique ID to a string.

Prototype

```c
int opsec_uuid_to_string(opsec_uuid *uuid, char *str_buf);
```
Arguments

Table 2-65  opsec_uuid_to_string arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>A pointer to a previously created Unique ID to convert to a string.</td>
</tr>
<tr>
<td>str_buf</td>
<td>An allocated array of chars to contain the converted string</td>
</tr>
</tbody>
</table>

Return Values

EO_OK upon success, EO_ERROR upon failure.

**opsec_uuid_from_string**

opsec_uuid_from_string converts a string to a Unique ID.

Prototype

```c
int opsec_uuid_from_string(opsec_uuid *uuid, const char *str);
```

Arguments

Table 2-66  opsec_uuid_from_string arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>A pointer to a previously created Unique ID to be converted from a string.</td>
</tr>
<tr>
<td>str</td>
<td>An allocated array of chars containing the string description of a UID.</td>
</tr>
</tbody>
</table>

Return Values

EO_OK upon success, EO_ERROR upon failure.
This section describes each of the OPSEC value API functions in detail. The function prototypes are defined in `opsec_vt_api.h`.

- **opsec_value_create**
  - **Prototype**: `opsec_value_t * opsec_value_create();`
  - **Arguments**: None.
  - **Return Values**: A pointer to the new OPSEC value if successful. `NULL` otherwise.

- **opsec_value_dest**
  - **Prototype**: `void opsec_value_dest (opsec_value_t *val);`
  - **Description**: Destroys an OPSEC value.
Arguments

Table 2-67 opsec_value_dest arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>val</td>
<td>A pointer to opsec_value_t.</td>
</tr>
</tbody>
</table>

Return Values

None.

**opsec_value_dup**

opsec_value_dup duplicates an existing OPSEC value.

Prototype

```c
opsec_value_t * opsec_value_dup (const opsec_value_t *val);
```

Arguments

Table 2-68 opsec_value_dup arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>val</td>
<td>A pointer to the OPSEC value which needs to be duplicated.</td>
</tr>
</tbody>
</table>

Return Values

A pointer to the new copy of the OPSEC value.

**Note** - The new duplicated opsec_value_t should be destroyed using opsec_value_dest

**opsec_value_set**

opsec_value_set sets an OPSEC value.

Prototype

```c
int opsec_value_set (opsec_value_t *val, OPSEC_VT type,...);
```
Arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>val</td>
<td>A pointer to the OPSEC value.</td>
</tr>
<tr>
<td>type</td>
<td>One of the OPSEC_VT types</td>
</tr>
</tbody>
</table>

Return Values

EO_OK on success, else EO_ERROR.

Note - opsec_value_set copies the value (including numbers, strings, buffers etc)
  - if value exists opsec_value_t will be destroyed
  - to set a buffer opsec_value_set expects the first argument to be the buffer length and then expects a pointer to the buffer.

For example suppose we have the following variables:

```c
char buf[10];
unsigned int buf_len = 10;
opsec_value_t *val;
```

Then the call should be:

```c
opsec_value_set(val, OPSEC_VT_BUFF, buf_len, buf);
```

**opsec_value_get**

opsec_value_get retrieves a value from opsec_value_t.

Prototype

```c
int opsec_value_get (const opsec_value_t *val, ...);
```
Arguments

Table 2-70  opsec_value_get arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>val</td>
<td>A pointer to opsec_value_t</td>
</tr>
<tr>
<td>...</td>
<td>An appropriately sized pointer to hold the value</td>
</tr>
</tbody>
</table>

Return Values

EO_OK on success, else EO_ERROR.

Note - If the out parameter is a string type, a reference to the value will be returned.

```c
cchar *str;
opsec_value_get(val, &str);
```

- If the out parameter is buffer type the first parameter is the address of a pointer (return reference to the value) and the second parameter is the size of the buffer

```c
char *buf;
unsigned int buf_len;
opsec_value_get(val, &buf, &buf_len);
```

**opsec_value_copy**

opsec_value_copy copies a value from one OPSEC value to another.

Prototype

```c
int opsec_value_copy (opsec_value_t *dst, const opsec_value_t *src);
```

Arguments

Table 2-71  opsec_value_copy arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>dst</td>
<td>A pointer to the target opsec_value_t.</td>
</tr>
<tr>
<td>src</td>
<td>A pointer to the source opsec_value_t.</td>
</tr>
</tbody>
</table>

Return Values

EO_OK on success, else EO_ERROR.

**opsec_value_get_type**

opsec_value_get_type returns the type that is stored in opsec_value_t.
Prototype

OPSEC_VT opsec_value_get_type (const opsec_value_t *val);

Arguments

Table 2-72  opsec_value_get_type arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>val</td>
<td>A pointer to opsec_value_t.</td>
</tr>
</tbody>
</table>

Return Values

The type that val holds (one of the OPSEC_VT types).
Table 2-73 lists OPSEC value types and their C equivalents.

<table>
<thead>
<tr>
<th>virtual type</th>
<th>C type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPSEC_VT_INT</td>
<td>int</td>
<td>32 bit</td>
</tr>
<tr>
<td>OPSEC_VT_UINT</td>
<td>unsigned int</td>
<td>general purpose unsigned integer</td>
</tr>
<tr>
<td>OPSEC_VT_I8BIT</td>
<td>int</td>
<td>8 bit signed integer</td>
</tr>
<tr>
<td>OPSEC_VT_UI8BIT</td>
<td>int</td>
<td>8 bit unsigned integer</td>
</tr>
<tr>
<td>OPSEC_VT_I16BIT</td>
<td>int</td>
<td>16 bit signed integer</td>
</tr>
<tr>
<td>OPSEC_VT_UI16BIT</td>
<td>int</td>
<td>16 bit unsigned integer</td>
</tr>
<tr>
<td>OPSEC_VT_I32BIT</td>
<td>int</td>
<td>32 bit signed integer</td>
</tr>
<tr>
<td>OPSEC_VT_UI32BIT</td>
<td>int</td>
<td>32 bit unsigned integer</td>
</tr>
<tr>
<td>OPSEC_VT_I64BIT</td>
<td>64 bits int</td>
<td>64 bit signed integer</td>
</tr>
<tr>
<td>OPSEC_VT_UI64BIT</td>
<td>64 bits int</td>
<td>64 bit unsigned integer</td>
</tr>
<tr>
<td>OPSEC_VT_HEX</td>
<td>unsigned int</td>
<td>general purpose hexadecimal number</td>
</tr>
<tr>
<td>OPSEC_VT_SHORT</td>
<td>short</td>
<td>general purpose short integer</td>
</tr>
<tr>
<td>OPSEC_VT_USHORT</td>
<td>unsigned short</td>
<td>general purpose unsigned short integer</td>
</tr>
<tr>
<td>OPSEC_VT_STRING</td>
<td>char *</td>
<td>general purpose string</td>
</tr>
<tr>
<td>OPSEC_VT_ISTRING</td>
<td>char *</td>
<td>case-insensitive string</td>
</tr>
<tr>
<td>OPSEC_VT_ACTION</td>
<td>int</td>
<td>the action taken in enforcing the Security Policy</td>
</tr>
<tr>
<td>OPSEC_VT_ALERT</td>
<td>int</td>
<td>the type of alert</td>
</tr>
<tr>
<td>OPSEC_VT_BUFF</td>
<td>char *</td>
<td>AMON character buffer</td>
</tr>
<tr>
<td>OPSEC_VT_DIRECTION</td>
<td>unsigned char</td>
<td>the direction of the connection (e.g. inbound)</td>
</tr>
<tr>
<td>OPSEC_VT_DURATION_TIME</td>
<td>unsigned int</td>
<td>the duration of the connection</td>
</tr>
<tr>
<td>OPSEC_VT_INTERFACE</td>
<td>int</td>
<td>the interface through which the connection is passed</td>
</tr>
</tbody>
</table>
## OPSEC Value Types

### Table 2-73: OPSEC value types

<table>
<thead>
<tr>
<th>virtual type</th>
<th>C type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPSEC_VT_IP</td>
<td>unsigned int</td>
<td>32 bit IP address in network order</td>
</tr>
<tr>
<td>OPSEC_VT_IP_PROTO</td>
<td>unsigned char</td>
<td>the IP protocol (TCP, UDP, etc.)</td>
</tr>
<tr>
<td>OPSEC_VT_IPV6</td>
<td>opsec_in6_addr</td>
<td>IPv6 address</td>
</tr>
<tr>
<td>OPSEC_VT_LOG</td>
<td>int</td>
<td>used by SAM</td>
</tr>
<tr>
<td>OPSEC_VT_MASK</td>
<td>unsigned int</td>
<td>mask (for UFP categories)</td>
</tr>
<tr>
<td>OPSEC_VT_PORT</td>
<td>unsigned short</td>
<td>16 bit port number</td>
</tr>
<tr>
<td>OPSEC_VT_NONE</td>
<td>—</td>
<td>empty value</td>
</tr>
<tr>
<td>OPSEC_VT_PROTO</td>
<td>short</td>
<td>the service (telnet, http, etc.)</td>
</tr>
<tr>
<td>OPSEC_VT_RPC_PROG</td>
<td>unsigned int</td>
<td>RPC program number</td>
</tr>
<tr>
<td>OPSEC_VT_RULE</td>
<td>int</td>
<td>the number of the Security Policy rule</td>
</tr>
<tr>
<td>OPSEC_VT_SR_HOSTGROUP</td>
<td>char *</td>
<td>group name (server-resolved) for LEA filter</td>
</tr>
<tr>
<td>OPSEC_VT_SR_HOSTNAME</td>
<td>char *</td>
<td>host name (server-resolved) for LEA filter</td>
</tr>
<tr>
<td>OPSEC_VT_SR_SERVICE</td>
<td>char *</td>
<td>service name (server-resolved) for LEA filter</td>
</tr>
<tr>
<td>OPSEC_VT_SR_SERVICEGR</td>
<td>char *</td>
<td>service group name (server-resolved) for LEA filter</td>
</tr>
<tr>
<td>OPSEC_VT_SR_USERGROUP</td>
<td>char *</td>
<td>group name (server-resolved) for LEA filter</td>
</tr>
<tr>
<td>OPSEC_VT_TIME</td>
<td>unsigned int</td>
<td>the time, in seconds, since 1 January, 1970</td>
</tr>
<tr>
<td>OPSEC_VT_TCP_PORT</td>
<td>unsigned short</td>
<td>TCP port number in network order</td>
</tr>
<tr>
<td>OPSEC_VT_UDP_PORT</td>
<td>unsigned short</td>
<td>UDP port number in network order</td>
</tr>
<tr>
<td>OPSEC_VT_UUID</td>
<td>opsec_uuid *</td>
<td>A Unique ID</td>
</tr>
</tbody>
</table>
Event Handlers

This section describes the functions that need to be written in order to implement an OPSEC Server or OPSEC Client. In two instances, events are client or server: OPSEC_SESSION_ESTABLISHED is a client only event and OPSEC_SERVER_FAILED_CONN_HANDLER is a server only event. See Table 1-9 on page 29 for a complete event handler server/client compatibility chart.

Event Handler for the OPSEC_SESSION_START event

This function is called when a new OPSEC session is created.

Note - The name StartHandler is a placeholder. You can assign any name to this function.

Prototype

int StartHandler (OpsecSession *session)

Arguments

Table 2-74 StartHandler (OPSEC_SESSION_START event) arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to the OpsecSession structure corresponding to the session.</td>
</tr>
</tbody>
</table>

Return Values

OPSEC_SESSION_OK if the session can continue.

OPSEC_SESSION_END if the session is to be closed.

OPSEC_SESSION_ERR if the session is to be closed due to an error

Notes

- In an OPSEC Client, StartHandler will be called before xxx_new_session returns. For example, in the following code fragment:

```c
sam_new_session(...);
printf(...);
```
StartHandler will be called before printf, so printf can use data that StartHandler initializes.

- It is recommended to avoid using API functions in StartHandler that involve communication between Client and Server. When StartHandler is called, the session between Client and Server is not completely initialized. Calling these API functions at this point may result in unexpected behavior.

To call API functions that involve communication between Client and Server from within StartHandler, proceed as follows:

- For a Server process, call opsec_schedule with the API function and OL as the interval. The function will be called after the session is initialized completely.
- For a Client process, call the API function after the session_create function is called.

Event Handler for the OPSEC_GENERIC_SESSION_START event

This function is called when a new OPSEC GENERIC session is created.

Note - The name StartHandler is a placeholder. You can assign any name to this function.

Prototype

int StartHandler (OpsecSession *session)

Arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to the OpsecSession structure corresponding to the generic session.</td>
</tr>
</tbody>
</table>

Return Values

- OPSEC_SESSION_OK if the session can continue.
- OPSEC_SESSION_END if the session is to be closed.
Event Handler for the OPSEC_SESSION_ESTABLISHED event

**OPSEC_SESSION_ERR** if the session is to be closed due to an error

**Notes**

- In an OPSEC Client, *StartHandler* will be called before *new_generic_session* returns. For example, in the following code fragment:
  ```c
  opsec_new_generic_session(...);
  printf(...);
  ```

  *StartHandler* will be called before *printf*, so *printf* can use data that
  *StartHandler* initializes.

- It is recommended to avoid using API functions in *StartHandler* that involve communication between Client and Server. When *StartHandler* is called, the session between Client and Server is not completely initialized. Calling these API functions at this point may result in unexpected behavior.

  To call API functions that involve communication between Client and Server from within *StartHandler*, proceed as follows:

  - For a Server process, call *opsec_schedule* with the API function and **OL** as the interval. The function will be called after the session is initialized completely.
  - For a Client process, call the API function after the *session_create* function is called.

**Event Handler for the OPSEC_SESSION_ESTABLISHED event**

This function is called when a new OPSEC session is completely established and acknowledged by the server

**Note** - The name *UserSessionEstablishedHandler* is a placeholder. You can assign any name to this function.

**Prototype**

```c
int UserSessionEstablishedHandler (OpsecSession *session)
```
Event Handler for the OPSEC_SESSION_END event

Arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to the OpsecSession structure corresponding to the session.</td>
</tr>
</tbody>
</table>

Return Values

OPSEC_SESSION_OK if the session can continue.

OPSEC_SESSION_END if the session is to be closed.

OPSEC_SESSION_ERR if the session is to be closed due to an error

Event Handler for the OPSEC_SESSION_END event

This function is called just before an OPSEC session ends.

Note - The name EndHandler is a placeholder. You can assign any name to this function.

This function should make no assumptions about what happened in the past, except that StartHandler was called. This event can occur at any point in the session, as a result of the OPSEC layer or another OPSEC entity terminating the session.

Note - When a session is terminated, the session pointer no longer points to valid data and should not be used. Any attempt to access or use the OpsecSession structure will result in undefined behavior.

After the session ends, any events that have not yet been processed should be descheduled (see “opsec_deschedule” on page 86). This prevents opsec_mainloop from processing these events using pointers that are no longer valid.

Prototype

void EndHandler (OpsecSession *session)
Event Handler for the OPSEC_GENERIC_SESSION_END event

Arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to the OpsecSession structure corresponding to the session.</td>
</tr>
</tbody>
</table>

Return Values

None.

Event Handler for the OPSEC_GENERIC_SESSION_END event

This function is called just before an OPSEC GENERIC session ends.

**Note** - The name EndHandler is a placeholder. You can assign any name to this function.

This function should make no assumptions about what happened in the past, except that StartHandler was called. This event can occur at any point in the session, as a result of the OPSEC layer or another OPSEC entity terminating the session.

**Note** - When a session is terminated, the session pointer no longer points to valid data and should not be used. Any attempt to access or use the OpsecSession structure will result in undefined behavior.

After the session ends, any events that have not yet been processed should be descheduled (see “opsec_deschedule” on page 86). This prevents opsec_mainloop from processing these events using pointers that are no longer valid.

Prototype

```c
void EndHandler (OpsecSession *session)
```
Event Handler for the OPSEC_SERVER_FAILED_CONN event

**Arguments**

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A pointer to the OpsecSession structure corresponding to the generic session.</td>
</tr>
</tbody>
</table>

**Return Values**

None.

**Event Handler for the OPSEC_SERVER_FAILED_CONN event**

This function is called if SIC fails while a connection is being established. `opsec_init_entity` (see page 69) assigns this function to handle the OPSEC_SERVER_FAILED_CONN event of a given Server entity. This function cannot be used with Client entities.

**Note** - The name `server_incomplete_handler` is a placeholder. You can assign any name to this function.

**Prototype**

```c
void server_incomplete_handler (OpsecEntity *entity, long peer_ip,
int sic_errno, char *sic_errmsg)
```
General Event Handlers

Arguments

Table 2-79  server_incomplete_handler (OPSEC_SERVER_FAILED_CONN event) arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>entity</td>
<td>A pointer to the Server entity.</td>
</tr>
<tr>
<td>peer_ip</td>
<td>The IP address of the VPN-1 Module acting as peer.</td>
</tr>
<tr>
<td>sic_errno</td>
<td>The error number set by the underlying SIC infrastructure. The meaning</td>
</tr>
<tr>
<td></td>
<td>of this number can be retrieved using opsec_get_sic_error.</td>
</tr>
<tr>
<td>sic_errmsg</td>
<td>A pointer to the corresponding SIC error message.</td>
</tr>
</tbody>
</table>

Return Values

None.

General Event Handlers

These are the event handlers to be used with the Events API. opsec_set_event_handler (see page 109) registers these functions to handle specific event instances.

Note that there are two kinds of data that are passed to an event instance and consequently to the event handler:

- data set when the event handler is defined (set_data)
- data set when the event is raised (raise_data)

Prototype

An event handler function is of type OpsecEventHandler, defined as follows:

```c
typedef int (*OpsecEventHandler)(OpsecEnv *env, long event, void *raise_data, void *set_data);
```
General Event Handlers

Arguments

Table 2-80  Event Handler Function arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>env</td>
<td>A pointer to the environment passed to opsec_mainloop.</td>
</tr>
<tr>
<td>event</td>
<td>The event identifier, as returned by opsec_new_event_id.</td>
</tr>
</tbody>
</table>
| raise_data | A pointer to the data set when the event is raised by
|            | opsec_raise_event (see page 112) or opsec_raise_persistent_event (see page 113). |
| set_data   | A pointer to the data set when the event is defined by
|            | opsec_set_event_handler (see page 109).                                |

Return Values

OPSEC_SESSION_OK if the session can continue.

OPSEC_SESSION_END if the session is to be closed.

OPSEC_SESSION_ERR if the session is to be closed due to an error.
THIRD PARTY TRADEMARKS AND COPYRIGHTS

Entrust is a registered trademark of Entrust Technologies, Inc. in the United States and other countries. Entrust’s logos and Entrust product and service names are also trademarks of Entrust Technologies, Inc. Entrust Technologies Limited is a wholly owned subsidiary of Entrust Technologies, Inc. FireWall-1 and SecuRemote incorporate certificate management technology from Entrust.

Verisign is a trademark of Verisign Inc.

The following statements refer to those portions of the software copyrighted by University of Michigan. Portions of the software copyright © 1992-1996 Regents of the University of Michigan. All rights reserved. Redistribution and use in source and binary forms are permitted provided that this notice is preserved and that due credit is given to the University of Michigan at Ann Arbor. The name of the University may not be used to endorse or promote products derived from this software without specific prior written permission. This software is provided “as is” without express or implied warranty. Copyright © Sax Software (terminal emulation only).

The following statements refer to those portions of the software copyrighted by Carnegie Mellon University.

Copyright 1997 by Carnegie Mellon University. All Rights Reserved.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of CMU not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.CMU DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL CMU BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

The following statements refer to those portions of the software copyrighted by The Open Group.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE OPEN GROUP BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

The following statements refer to those portions of the software copyrighted by The OpenSSL Project. This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/).

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT "AS IS" AND ANY * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The following statements refer to those portions of the software copyrighted by Eric Young. THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. Copyright © 1998 The Open Group.
The following statements refer to those portions of the software copyrighted by Jean-loup Gailly and Mark Adler. Copyright (C) 1995-2002 Jean-loup Gailly and Mark Adler. This software is provided "as-is", without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software. Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

3. This notice may not be removed or altered from any source distribution.

The following statements refer to those portions of the software copyrighted by the Gnu Public License. This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Boston, MA 02110-1301, USA.

The following statements refer to those portions of the software copyrighted by others. The copyright holder disclaims all warranties, either express or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Boston, MA 02110-1301, USA.

The curl license

COPYRIGHT AND PERMISSION NOTICE

Copyright (c) 1996 - 2004, Daniel Stenberg, <daniel@haxx.se>. All rights reserved.

Permission to use, copy, modify, and distribute this software for any purpose
with or without fee is hereby granted, provided that the above copyright
notice and this permission notice appear in all copies.
Chapter 137

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

The PHP License, version 3.0

Copyright (c) 1999 - 2004 The PHP Group. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name "PHP" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact group@php.net.

4. Products derived from this software may not be called "PHP", nor may "PHP" appear in their name, without prior written permission from group@php.net. You may indicate that your software works in conjunction with PHP by saying "Foo for PHP" instead of calling it "PHP Foo" or "phpfoo".

5. The PHP Group may publish revised and/or new versions of the license from time to time. Each version will be given a distinguishing version number. Once covered code has been published under a particular version of the license, you may always continue to use it under the terms of that version. You may also choose to use such covered code under the terms of any subsequent version of the license published by the PHP Group. No one other than the PHP Group has the right to modify the terms applicable to covered code created under this License.

6. Redistributions of any form whatsoever must retain the following acknowledgment:

   "This product includes PHP, freely available from <http://www.php.net/>".

THIS SOFTWARE IS PROVIDED BY THE PHP DEVELOPMENT TEAM "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE PHP DEVELOPMENT TEAM OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the PHP Group. The PHP Group can be contacted via Email at group@php.net.

For more information on the PHP Group and the PHP project, please see <http://www.php.net>. This product includes the Zend Engine, freely available at <http://www.zend.com>.

This product includes software written by Tim Hudson (tjh@cryptsoft.com).

Copyright (c) 2003, Itai Tzur <itzur@actcom.co.il>

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistribution of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Neither the name of Itai Tzur nor the names of other contributors may be used to endorse or promote products derived from this software without specific prior written permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
THE POSSIBILITY OF SUCH DAMAGE.

Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to
the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of
the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT
LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN
NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Copyright © 2003, 2004 NextHop Technologies, Inc. All rights reserved.

Confidential Copyright Notice

Except as stated herein, none of the material provided as a part of this document may be copied, reproduced, distrib-uted,
republished, downloaded, displayed, posted or transmitted in any form or by any means, including, but not lim-it-ed to, electronic,
mechanical, photocopying, recording, or otherwise, without the prior written permission of NextHop Technologies, Inc. Permission is
granted to display, copy, distribute and download the materials in this doc-ument for personal, non-commercial use only, provided you
do not modify the materials and that you retain all copy-right and other proprietary notices contained in the materials unless otherwise
stated. No material contained in this document may be "mirrored" on any server without written permission of NextHop. Any
unauthorized use of any material contained in this document may violate copyright laws, trademark laws, the laws of privacy and
publicity, and communications regulations and statutes. Permission terminates automatically if any of these terms or condi-tions are
breached. Upon termination, any downloaded and printed materials must be immediately destroyed.

Trademark Notice

The trademarks, service marks, and logos (the "Trademarks") used and displayed in this document are registered and unregistered
Trademarks of NextHop in the US and/or other countries. The names of actual companies and products mentioned herein may be
Trademarks of their respective owners. Nothing in this document should be construed as granting, by implication, estoppel, or
otherwise, any license or right to use any Trademark displayed in the document. The owners aggressively enforce their intellectual
property rights to the fullest extent of the law. The Trademarks may not be used in any way, including in advertising or publicly
pertaining to distribution of, or access to, materials in

this document, including use, without prior, written permission. Use of Trademarks as a "hot" link to any website is prohibited unless
establishment of such a link is approved in advance in writing. Any questions concerning the use of these Trademarks should be
referred to NextHop at U.S. +1 734 222 1600.

U.S. Government Restricted Rights

The material in document is provided with "RESTRICTED RIGHTS." Software and accompanying documentation are provided to the
U.S. government ("Government") in a transaction subject to the Federal Acquisition Regulations with Restricted Rights. The
Government's rights to use, modify, reproduce, release, perform, display or disclose are

restricted by paragraph (b)(3) of the Rights in Noncommercial Computer Software and Noncommercial Computer Soft-ware
Documentation clause at DFAR 252.227-7014 (Jun 1995), and the other restrictions and terms in paragraph (g)(3)(i) of Rights in
Data-General clause at FAR 52.227-14, Alternative III (Jun 87) and paragraph (c)(2) of the Commer-cial
Computer Software-Restricted Rights clause at FAR 52.227-19 (Jun 1987).

Use of the material in this document by the Government constitutes acknowledgment of NextHop's proprietary rights in them, or that of
the original creator. The Contractor/Licensor is NextHop located at 1911 Landings Drive, Mountain View, California 94043. Use,
duplication, or disclosure by the Government is subject to restrictions as set forth in applicable laws and regulations.
Disclaimer Warranty Disclaimer Warranty Disclaimer Warranty Disclaimer Warranty

THE MATERIAL IN THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTIES OF ANY KIND EITHER EXPRESS OR IMPLIED. TO THE FULLEST EXTENT POSSIBLE PURSUANT TO THE APPLICABLE LAW, NEXTHOP DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT OR OTHER VIOLATION OF RIGHTS. NEITHER NEXTHOP NOR ANY OTHER PROVIDER OR DEVELOPER OF MATERIAL CONTAINED IN THIS DOCUMENT WARRANTS OR MAKES ANY REPRESENTATIONS REGARDING THE USE, VALIDITY, ACCURACY, OR RELIABILITY OF, OR THE RESULTS OF THE USE OF, OR OTHERWISE RESPECTING, THE MATERIAL IN THIS DOCUMENT.

Limitation of Liability

UNDER NO CIRCUMSTANCES SHALL NEXTHOP BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, INCLUDING, BUT NOT LIMITED TO, LOSS OF DATA OR PROFIT, ARISING OUT OF THE USE, OR THE INABILITY TO USE, THE MATERIAL IN THIS DOCUMENT, EVEN IF NEXTHOP OR A NEXTHOP AUTHORIZED REPRESENTATIVE HAS ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IF YOUR USE OF MATERIAL FROM THIS DOCUMENT RESULTS IN THE NEED FOR SERVICING, REPAIR OR CORRECTION OF EQUIPMENT OR DATA, YOU ASSUME ANY COSTS THEREOF. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT FULLY APPLY TO YOU.

Copyright © ComponentOne, LLC 1991-2002. All Rights Reserved.

BIND: ISC Bind (Copyright (c) 2004 by Internet Systems Consortium, Inc. ("ISC"))

Copyright 1997-2001, Theo de Raadt: the OpenBSD 2.9 Release

PCRE LICENCE

PCRE is a library of functions to support regular expressions whose syntax and semantics are as close as possible to those of the Perl 5 language. Release 5 of PCRE is distributed under the terms of the "BSD" licence, as specified below. The documentation for PCRE, supplied in the "doc" directory, is distributed under the same terms as the software itself.

Written by: Philip Hazel <ph10@cam.ac.uk>

University of Cambridge Computing Service, Cambridge, England. Phone:
+44 1223 334714.

Copyright (c) 1997-2004 University of Cambridge All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of the University of Cambridge nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
Index

A

APIs
  Multithread
    Level 33
asym_sslca
  API configuration file
    parameter 48
  fwopsec.conf parameter 45
asym_sslca_comp
  API configuration file
    parameter 48
  fwopsec.conf parameter 45
asym_sslca_rc4
  API configuration file
    parameter 48
  fwopsec.conf parameter 45
asym_sslca_rc4_comp
  API configuration file
    parameter 48
  fwopsec.conf parameter 45
auth_opsec
  API configuration file
    parameter 45, 48
auth_port
  API configuration file
    parameter 45
auth_type
  and SIC 54
  API configuration file
    parameter 45, 47, 49
authenticated connections
  description 37, 38
  key exchange 40

C

certificates 37
clear connections
  configuring OPSEC Client
    application 49
    description 37, 38
client 19

D

data structures
  OpsecEntity 25
  OpsecEnv 25
  OpsecInfo 27
  OpsecSession 25
debugging 24
  OPSEC_DEBUG_LEVEL
    environment variable 24
    output to stderr 24

E

encrypted connections
  authentication with certificates 37
  description 37, 38
  key exchange 40
  environment variables 22
  OPSEC_DEBUG_LEVEL 24
  OPSECDIR 23

errors
  converting OPSEC error codes to strings 106
  converting SIC error codes to strings 106

events
  API functions 36
  handlers 34
  identifying 34
  persistent 35
  regular 35
  scheduled 34
  sockets 34
  types 35

F

fwnl
  API configuration file
    parameter 45, 48

g

generic OPSEC session 21

H

host
  API configuration file
    parameter 45

I

include files
  opsec.h 9, 60, 105
  opsec_error.h 106
  opsec_event.h 108
  opsec_uid.h 116

ip
API configuration file parameter 45, 47
IP binding 47

K
keep-alive service 90
key exchange 40
authenticated connections 40
encrypted connections 40

L
local connections description 38

M
macros
  general OPSEC macros 105
method
  API configuration file parameter 48
multiplexing mode 32, 72
Multithread
  Reentrant APIs 33
Thread-Safe APIs 33
multithreading 33

N
none
  API configuration file parameter 45

O
opsec_errno global variable 24
OPSEC
  common functions 19
  OPSEC Client configuration 49
OPSEC elements
  relationship among 21
OPSEC entity
definition 21
OPSEC environment
definition 21
OPSEC Server configuration 47
OPSEC service-specific
  session 25
OPSEC session
definition 21
  generic 21
  service-specific 21, 25
OPSEC Unique ID 32
OPSEC unique IDs 116
opsec.h 9, 60, 105, 106
OPSEC_CONF_ARGV 49
OPSEC_CONF_ARGV attribute 61
OPSEC_CONF_FILE 45
OPSEC_CONF_FILE attribute 61
OPSEC_DEBUG_LEVEL environment variable 24
opsec_del_event_handler 109
opsec_del_socket_event 88
opsec_deschedule 86
opsec_destroy_entity 74
opsec_DllMain 91
opsec_end_session 78
opsec_end_session_reason 78
OPSEC_ENTITY_SIC_NAME 44
opsec_entity_sic_name
  API configuration file parameter 46
opsec_env_destroy 64
OPSEC_EOL attribute 63
opsec_errno 34
  global variable 106
opsec_errno_ptr 106
opsec_error.h 106
opsec_event.h 108, 116
opsec_free 91
OPSEC_GENERIC_SESSION_END
  D 30
OPSEC_GENERIC_SESSION_START 30
OPSEC_GENERIC_SESSION_START_HANDLER 31
opsec_get_conf 67
opsec_get_local_address 97
opsec_get_my_sic_name 102
opsec_get_own_entity 80
opsec_get_peer_address 98
opsec_get_peer_entity 80
opsec_get_peer_sdk_version 96
opsec_get_sdk_version 96
opsec_get_session_env 80
opsec_get_sic_error 106
opsec_info_destroy 82
opsec_info_get 83
opsec_info_init 81
opsec_info_set 82
opsec_init 60
opsec_init_entity 31, 32, 69
  and SIC 54
opsec_israised_event 114
opsec_mainloop 84
  terminating 36
OPSEC_MT attribute 62
opsec_new_event_id 108
opsec_ping 92
opsec_pull_cert 42
opsec_putkey 40
opsec_raise_event 35, 112
opsec_raise_persistent_event 35, 113
opsec_resume_event_handler 11
  1
opsec_resume_session_read 77
opsec_schedule 85, 86
OPSEC_SERVER_FAILED_CONN 30, 31
OPSEC_SESSION_END 30, 31
OPSEC_SESSION_ESTABLISHED 30, 31
OPSEC_SESSION_START 30, 31
opsec_set_event_handler 109
opsec_set_session_timeout 94
opsec_set_session_timeout_handler 95
opsec_set_socket_event 87
OPSEC_SHARED_LOCAL_PATH attribute 62
opsec_sic_get_peer_cert_hash 10
  3
opsec_sic_get_peer_sic_name 10
  2
opsec_sic_get_sic_method 103
OPSEC_SIC_NAME 43
opsec_sic_name
API configuration file parameter 46
OPSEC_SIC_NAME attribute 61
OPSEC_SIC_POLICY_FILE attribute 61
OPSEC_SSLCA_BUFFER 61, 62
OPSEC_SSLCA_FILE 61
opsec_start_keep_alive 90
opsec_start_server 76
opsec_stop_keep_alive 90
opsec_stop_server 76
opsec_suspend_event_handler 110
opsec_suspend_session_read 77
opsec_unraise_event 114
opsec_uuid.h 116
opsec_uuid_create 116
opsec_uuid_destroy 116
opsec_uuid_duplicate 117
opsec_uuid_equal 118
opsec_uuid_from_string 119
opsec_uuid_set_unspecified 117
opsec_uuid_to_string 118
opsec_value_copy 123
opsec_value_create 120
opsec_value_dest 120
opsec_value_dup 121
opsec_value_get 122
opsec_value_get_type 123
opsec_value_set 121
OPSECDIR environment variable 23
OpsecEntity data structure 25
OpsecEnv 33
OpsecEnv data structure 25
OpsecInfo data structure 27
OpsecSession data structure 25
general OPSEC functions 9, 60

R
regular event description 35

S
scheduled events 34
Secure Internal Communication infrastructure, see SIC
server 19
service-specific OPSEC session 21
SESSION_OPAQUE 27, 105
SIC 19, 37
configuration precedence 54
handling failure 132
initialing infrastructure 43
policy file 50
SIC name 61
definition 43
SIC policy file 43
socket events 34
ssl
API configuration file parameter 45, 48
SSL connections. See encrypted connections
ssl_clear
API configuration file parameter 48
ssl_clear_opsec
API configuration file parameter 45, 48
ssl_opsec
API configuration file parameter 48
API configuration file wopsec.conf parameter 45
sslca
API configuration file parameter 45, 48
sslca_clear

P
parameters of f 97
persistent events 35
policy file
SIC 43
port
API configuration file parameter 45, 47, 48, 49
prototypes

T
TCP
keep-alive service 90
multiplexing mode 32, 72
threads
multithreading 33
Thread-Safe
Events APIs 33
transport layer 19

V
version
OPSEC 96