Intel® Active Management Technology System Defense Feature and Agent Presence Overview

Contents

Overview

The System Defense Feature (Circuit Breaker or CB) and Agent Presence (AP) are security toolsets built into Intel® Active Management Technology (Intel® AMT). The System Defense Feature and Agent Presence toolsets are targeted at closing two gaps in the Intrusion Detection System (IDS) methods currently employed by IT:

- The time window between the identification of an OS/Agent vulnerability and completed deployment of a corresponding patch.
- The time between an end-user tampering with an IDS agent and detection of and acting on the tampering.

The System Defense Feature toolset allows a Management Console (MC) application to define and enforce network security policies. A System Defense Feature policy contains a set of filters that are applied to incoming and outgoing network packets combined with actions to take when a packet matches the conditions in the filter or does not match the conditions in the filter. System Defense Feature policies are loaded onto a platform containing an Intel AMT device by a Management Console application. Once a System Defense Feature policy is activated, the Intel AMT device inspects each incoming and outgoing packet and performs the necessary action specified in the policy.

The Agent Presence toolset enables MC applications to configure Intel AMT devices to monitor for the presence of software agents such as Anti-Virus and Firewall applications running on the Intel AMT system platform. The MC application configures the Intel AMT device with timers set to detect when the software agent initializes and periodically transmits "heartbeat" signals. If any of the timers expire, AP will perform an action. Possible actions include one or all of the following:

- Activate a pre-programmed System Defense Feature policy which contains Network Isolation filters.
- Send an SNMP platform event trigger (PET) alert.
- Log the event to the local Intel AMT event log.

For detailed information about System Defense Feature and Agent Presence functions and data structures, see the *Intel® Active Management Technology Network Interface Guide*.

The System Defense Feature was formerly called “Circuit Breaker”. The *Network Interface Guide* refers to the Circuit Breaker Interface and Circuit Breaker Realm. The functions and data structures associated with the toolset all have “CB” as part of their names. The toolset and functions will continue to be called Circuit Breaker or CB in this document.

Audience and Prerequisites

This document provides application developers with the information necessary to understand and make use of the Intel AMT System Defense Feature (Circuit Breaker) and Agent Presence (AP) toolsets. Readers need to have an understanding of TCP/IP and Ethernet protocols, networking, and network security.

Terms and Acronyms

Acronym or Term	Definition
Agent	Software that runs on a client PC with OS running.
AP	Agent Presence
AV	Anti-Virus software agent
Alert	An alert is sent when the Intel AMT firmware (FW) notifies the Management Console that an event occurred in the system. An event can be physical occurrence, such as a fan failure or a filter-detected occurrence, such as a virus attack
CB	Circuit Breaker, the previous name for the System Defense Feature
GUID	Global Unique Identifier, a 16-character identifier used to identify an Agent to Intel AMT AP.
Host or Host CPU	The processor that is running the operating system. This is separate from the embedded processor running the Intel AMT FW
Host Service/Application	An application that runs on the host CPU.
IANA	Internet Assigned Numbers Authority (see http://www.iana.org).
Intel AMT FW	The Intel AMT firmware that runs on the embedded processor.
ISV	Independent Software Vendor
IT User	Information Technology staff member that uses the Management Console to oversee multiple PCs on a network.
MC (Management Console)	Centralized software that communicates with Intel AMT. It is used to manage multiple PCs
NVM	Non-Volatile Memory - A memory that will not have its content erased even if there is no power applied to it. Intel AMT uses a FLASH device for NVM.
OOB interface	Out Of Band interface. This is the SOAP over HTTP or SOAP over HTTPS protocol.
PET	Platform event trap – a specification (see the Intel AMT Network Interface Guide) that defines the format used by managed systems to alert a remote console.
Remote Management application	An application running on a Management Console that sends commands and configurations to the Intel AMT FW via the OOB interface
System States	Operating System power states such as S0.

Roles and Responsibilities

Circuit Breaker and Agent Presence provide powerful features that, if they are improperly configured, could result in undesirable effects and could impact the overall usability of the Intel AMT-enabled system platform.

Intel recommends the following practices:

1. Be sure you fully understand IP networking, network administration, security and Intel AMT Circuit Breaker and Agent Presence.
2. Define simple and traceable security scenarios. See the Use Cases described below for examples.
3. Give Policies, Filters, and Agents meaningful names.
4. Define and write a security policy document, review this document and every possible scenario with experts.
5. Test all defined scenarios to ensure that you see the expected results.

ISV Applications

ISV applications can be categorized as a Management Console or as a Local Agent. The following paragraphs list the functions of ISV applications that interact with Intel AMT to use the CB and AP functionality.

Management Console (MC)

A Management Console application supplies a GUI / Web tool that enables IT personnel to:

1. Define CB filters and policies.
2. Define AP watchdog monitoring and actions.
3. Discover Intel AMT devices.
4. Apply CB filters, policies, AP monitoring and actions to Intel AMT devices.
5. Enable operators to manually enable ad-hoc CB policies.
6. Manage events by:
    a. Polling events from Intel AMT devices event logs.
    b. Processing Intel AMT events and alerts by displaying them, forwarding alerts to IT personnel, or automatically applying CB to respond to the alerts.

Local Agent

An agent running on an Intel AMT-enabled platform interacts with the local Intel AMT firmware in the following ways:

1. Registers to Intel AMT Agent Presence watchdog services upon initialization.
2. Sends heartbeats to Intel AMT.
3. Reports shutdown to Intel AMT upon termination.

Staff Responsibilities

Security planning and implementation are essential for successfully deploying a CB and AP application. The following lists the responsibilities of enterprise staff in planning and executing use of CB and AP.

Enterprise Security Officer

Responsibilities:

1. Define enterprise security policy.
2. Define CB and AP plans.
3. Monitor and Apply ad-hoc CB policies to handle security risks.
4. Define mitigation rules, to handle possible security alerts.

IT Administrator

Responsibilities:

1. Use MC to define CB and AP as outlined by an enterprise security officer.
2. Monitor events and alerts from Intel AMT CB and AP.

Help Desk Operator

Responsibilities:

1. Use MC to monitor and handle security events and alerts.
2. Apply security mitigations, defined by IT administrator and security officer.

Circuit Breaker (CB)

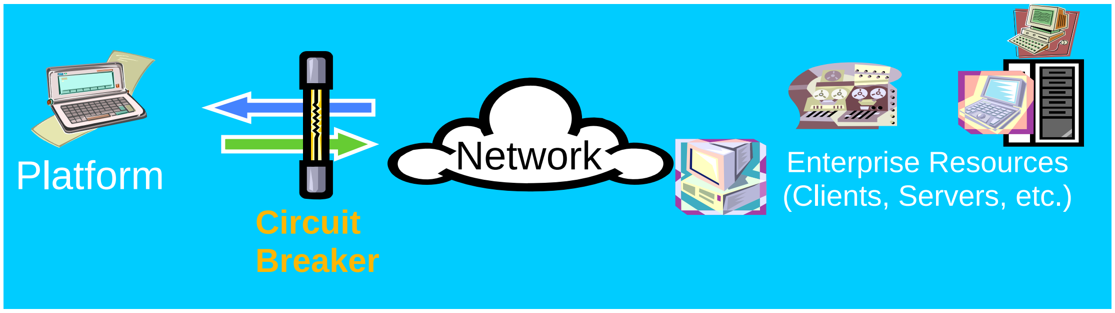


Figure 1: Circuit Breaker

Circuit Breaker is a set of capabilities that allows selective network isolation of Ethernet and IP protocol flows based on policies set by a remote Management Console. The targeted applications include Anti-Virus management frameworks and Intrusion Detection Systems (IDS). Circuit Breaker policies may only be set over the network interface by remote Management Consoles, but not by a Local Agent.

Tamper-resistance and system-state independence were two key design goals for Circuit Breaker. The Circuit Breaker capability is highly resistant to attack from mal-ware, and provides network isolation capabilities regardless of the operating state of the OS.

Network Isolation

The Circuit Breaker Network Isolation capabilities are based on a set of packet filters that are applied to both in-bound and out-bound packet streams. These filters allow the Management Console to pass or block specific IP-based network flows and to keep traffic counts or log the occurrence of these flows. Intel AMT supports 32 in-bound (Rx) and 32 out-bound (Tx) filters. One each of the 32 Tx and Rx filters is used as the “else” (non-matching) filter. If anti-spoofing is enabled, it utilizes one of the Tx filters. This reduces the available filters to 31 in-bound and 30 out-bound.

The filters support bit-level masking of IP Source and Destination Addresses and support ranges on Source and Destination Ports.

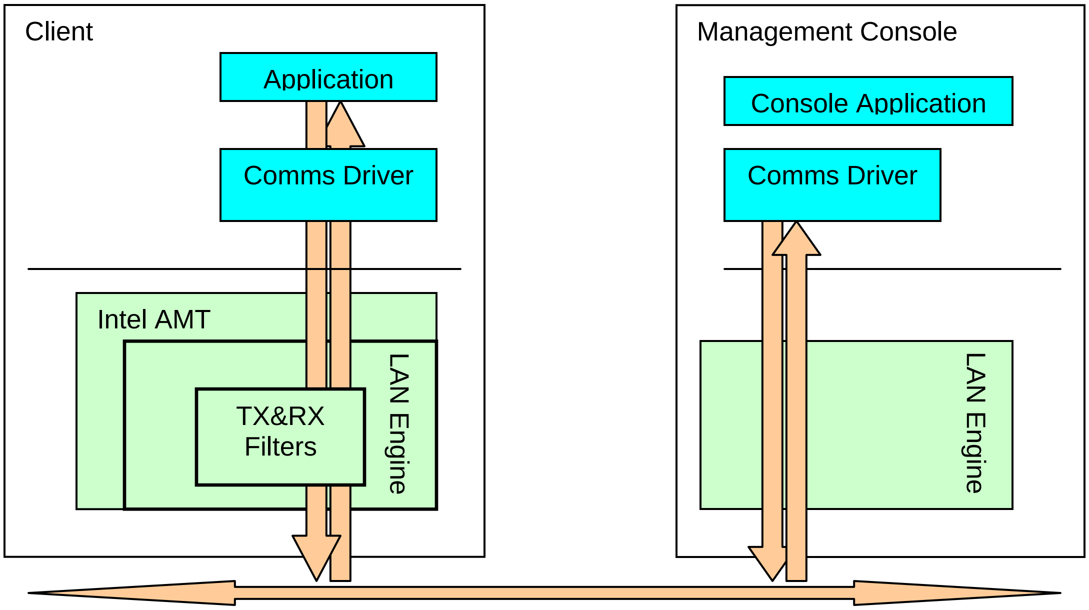


Figure 2: Network Isolation Overview

Intel AMT with Circuit Breaker can block or isolate the Client PC from specific TCP/IP flows. The isolation mechanism is a set of transmit and receive filters.

Figure 2: Network Isolation Overview shows the overall Circuit Breaker System.

As shown in the diagram, the Client PC contains the following components:

1. The “LAN engine” device that provides access to the network. For Intel AMT it is an Ethernet connection to the network.
2. The “TX & RX Filters” are Transmit and Receive filters.

3. Intel AMT is the only component that is able to set/modify the configuration of the “TX & RX Filters” via requests from the Management Console.
4. The “COMMS Driver” is the network driver running in the context of the Client OS. This driver is considered not trusted and has no access to the filter configuration.
5. “Application” is an application or service running in the OS context, and using the network for communication.

The system operates as follows:

1. The Management Console connects to Intel AMT over the network through a secure Out-of-Band channel. The Management Console sets Circuit Breaker policies used by Intel AMT to control the configuration of the “TX & RX Filters”.
2. Intel AMT selects the Active Policy based on the Policy precedence.
3. Intel AMT activates the filters associated with the Active Policy.
4. Each packet sent or received by client Applications passes through the “TX & RX Filters” allowing the Circuit Breaker filters to isolate specific flows.

CB Policies and Filters

A CB policy is a set of CB filters. A policy contains a subset of all the defined CB filters. The filters associated with a policy are enabled when this policy becomes the Active Policy. The policy structure created by an MC application contains the following members:

- Policy name (optional)
- Policy Precedence
- AntiSpoofingFilter enable/disable
- A list of filters which are enabled when the policy is active. (Note: The maximum number of transmit and receive filters in a policy is returned from `CbQueryCapabilities`)
- Default Tx filter. (see Default Filter)
- Default Rx filter. (see Default Filter)

Anti Spoofing

Spoofing is a term used for a host trying to falsify its identity by sending IP packets with a source IP address different from its assigned IP address. Intel AMT implements anti-spoofing by checking all outgoing packets, and comparing the source IP to the network interface IP address. If the IP addresses do not match, the packets will be dropped. Anti spoofing uses a transmit filter. If this feature is enabled, the available transmit filters are decreased from 31 to 30, when the “else” filters are taken into account. Anti spoofing is an option in a Circuit Breaker policy.

Active Policy

For each network interface that supports Circuit Breaker:

1. There is at most one **Active** policy.
2. There are at most two **Enabled** policies:

   a. One enabled policy is set by the Management Console (MC). This may not exist if no policy was enabled by the MC.

   b. One enabled policy is set by the Agent Presence toolset - see _Agent_Presence_(AP) This entry may be empty if no policy was enabled by AP.

3. The Active Policy is the enabled policy with the higher precedence.

4. An MC enables a policy in the following way:

   a. The MC calls `CbPolicyEnable(EnablePolicies, ActivePolicies).`

   b. `EnablePolicies`: is an input parameter, containing an array of `CircuitBreakerHardwarePolicyType`, which MC wants to enable. Each entry in this array associates a CB policy to a network interface. If there is only one network interface, there will be only one policy in the list. `CircuitBreakerHardwarePolicyType` contains 2 members:

      - `HardwareID:` Identifier of the network interface

      - `PolicyCreationHandle:` Identifier of the policy to enable.

   c. `ActivePolicies:` is an output parameter, containing an array of `CircuitBreakerHardwarePolicyType.` Each entry in the array specifies the Active policy for a network interface.

Note: Calling `CbPolicyEnable()` causes the disabling of the previous policy. For example, If the MC application calls `CbPolicyEnable()` to enable policy1, and subsequently calls `CbPolicyEnable()` to enable policy2, policy2 is enabled. policy1 will not remain enabled or be an active policy.

1. To disable a policy call `CbPolicyDisable()`.

2. To get the Active policy for a network interface, call `CbPolicyGetEnabled`() to get a list of enabled policies. Call `CbPolicyGetActiveStatistics`() to identify the active policy.

Filters

A Management Console defines a filter by calling `CbFilterCreate (CircuitBreakerFilterType, FilterCreationHandle)`.

This function receives an input parameter `CircuitBreakerFilterType` – a structure containing the properties of the new filter, and returns a handle to the newly created filter.

The filter properties include:

- `FilterName`: Filter name (optional)
- `FilterDirection`: Transmit or Receive packets.
- `FilterProfile`: Determines whether the filter action is Pass With Statistics, Drop With Statistics, Drop, Pass or Rate limit filter (see below).
- `FilterProfileData`: for Rate Limit only: the maximum number of events transmitted or received per second.
- `FilterPacket` : Describes the packet type to be filtered: Ethernet, IP, UDP or TCP.

- `ActionEventOnMatch`: States whether an event should be raised via the Intel AMT event manager.

See Appendix A - Networking Packet Structures for protocol specific filters.

Default Filter

A default “Else” filter, for both receive and transmit directions, is available for catching all packets not matched to any of the policy filters.

Transmit Filters

Transmit filters are applied to packets transmitted from the host client PC to the network. Such filters can be used to block all traffic from a host suspected as infected by a mal-ware, from hurting other hosts or the network

Receive Filters

Receive filters are applied to packets received from the network by the host client PC. Such filters can be used to block all packets received by the host after boot until an antivirus agent starts.

IPV4 and IPV6

The Management Console can define both IPV4 filters and IPV6 filters. Since an IPV6 address length is 16 bytes, four of the 31 filter entries in either direction are required for one IPv6 address. The maximum number of IPV6 filters in a policy is 7 transmit and 7 receive filters.

Statistical Filters

The MC application can use a statistical filter, for collecting statistical data. Intel AMT counts the number of packets that match the condition in the filter. The MC application can read these counters, and reset them.

To define a statistical counter Pass filter call `CbFilterCreate() with CircuitBreakerFilterType.FilterProfile = FilterProfileStisticsPass.`

To define a statistical counter Drop filter call `CbFilterCreate() with CircuitBreakerFilterType.FilterProfile = FilterProfileStisticsDrop.`

To read these statistics counters, call `CbPolicyGetActiveStatistics ().`

Rate Limit Filters

The MC application can define rate limit filters that limit the number of specific types of packets per second received or transmitted. A Rate Limit filter behaves like a statistics filter with a threshold in that it counts events like a statistics filter, but it has the additional action of cutting off traffic if the threshold is reached. To define a Rate Limit filter call `CbFilterCreate()` with `CircuitBreakerFilterType.FilterProfile = FilterProfileRateLimit` and `Filter.FilterProfileData` = THRESHOLD (maximum number of packets per second).

Each second the Rate Limit filter allows matching packets to pass until the threshold number is reached and blocks all other matching packets for the remainder of the second. For example the IT manager can specify a filter limiting the number of SYN packets per second sent from the host to the network.

The maximum number of Tx and Rx statistical filters and Rate Limit filters combined in a policy is 16 (IPv4) or 4 (IPv6). Note also that the `CbPolicyGetActiveStatistics` function does not return the number of packets that match the filter condition in a Rate Limit filter.

Filter Actions

When a packet matches the conditions in a filter the following actions may take place:

- If the filter is a Drop filter, the packet is discarded. It is not sent to the host driver or to the network.
- If the filter is a rate limit filter and the number of packets in the last second exceeds the threshold, the packet is dropped.
- If an event is defined for this filter, an event is raised by the Intel AMT event manager. MC applications can register with the Intel AMT device to receive PET alerts on these events and/or store the events in the Intel AMT event log. An event is raised only once per filter until the next call to `CbGetActiveStatistics` or in the event of a power down.

    - To define an event set the `ActionEventOnMatch` field in the `CircuitBreakerFilterType` structure, used to create the event in `CbFilterCreate().`
    - If the filter is Rate Limit, no more then one event is generated each second.

Determining Circuit Breaker Capabilities

Use `CbQueryCapabilities()` to retrieve the circuit breaker capabilities of each network interface. These capabilities include: maximum number of filters and counters, maximum number of transmit and receive filters in a policy, for both IPV4 and IPV6, and various capabilities and limitations of Circuit Breaker associated with this interface.

How Networking Packets Are Processed by CB filters

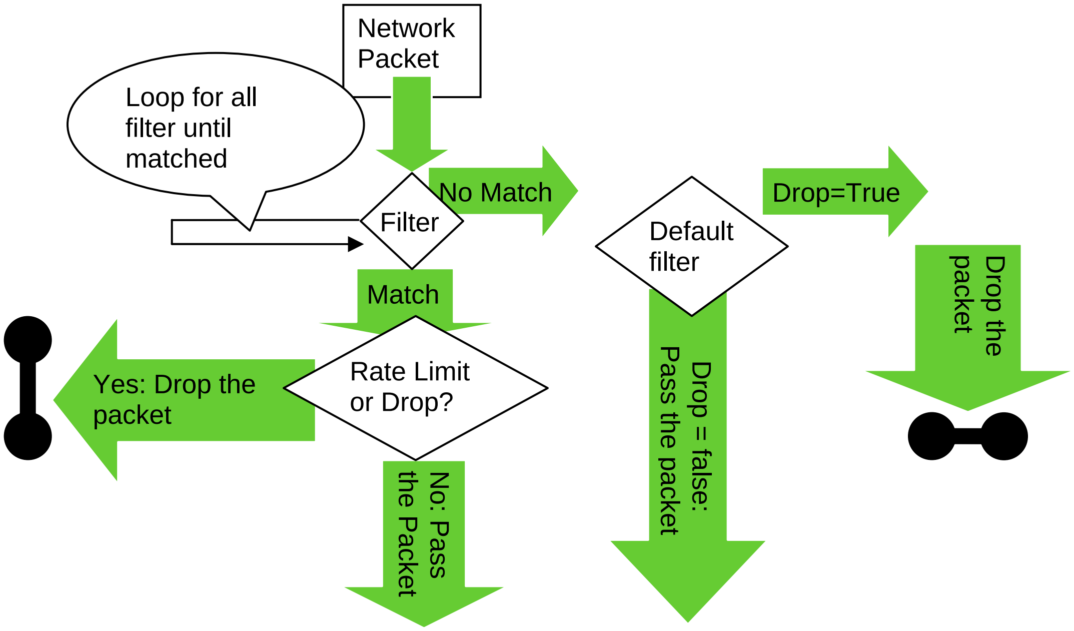


Figure 3 Network Filter Matching

Figure 3, illustrates the processing flow of a packet processed by CB filters.

For each received / transmitted packet:

- Check with each filter in active policy
- If no filter match, check default filter:
  - If Drop: Drop the packet.
  - Else: Allow the packet to pass to the OS driver or to the network.
- If any filter matched:
  - If one of the matched filters is a drop filter or a rate limit filter that has reached its threshold, drop the packet.
  - Else, Pass the packet

Filtering DHCP Transmit Traffic

When Intel AMT and the host CPU are configured to use DHCP for IP address assignment, the host communicates with a DHCP server using the DHCP protocol. By default, Intel AMT does not block DHCP transmit traffic. There is a built-in Pass filter for this traffic that is included in any active policy, so even if a policy has a default Block filter intended to block all outgoing traffic, the DHCP transmit packets will not be blocked.

To block DHCP transmit traffic, add an additional filter to the Circuit Breaker policy that will explicitly do so. As shown in Figure 3, above, if a packet matches a Drop filter, it will be dropped, even if it also matches a Pass filter. A “Drop DHCP transmit packets” filter has the following characteristics:

• Filter Direction:	Transmit
• Filter Profile:	Drop
• Protocol:	UDP
• Source Port:	0x4400
• Destination Port:	0x4300

CB and Machine State

All the CB and Agent Presence definitions are stored in non volatile memory, and are preserved regardless of the host and Intel AMT system and power state.

Agent Presence (AP)

ISVs produce a large number of products that run within the OS context and offer management services to Enterprise IT departments. Among the products being offered are asset tracking, application monitoring, system performance monitoring and provisioning, intrusion detection systems and local firewalls. These products are installed using an agent/console model where the agent executes on the local client and communicates with a Management Console application that runs on a machine located elsewhere in the network. Unfortunately it is not difficult for the local user to compromise the agent, either by killing the process or stopping the service.

The Intel AMT Agent Presence (AP) toolset monitors the presence of such software agents. AP takes specific actions if it detects an agent is no longer active. The actions that are possible when Agent is not present include:

- Changing the configuration of the Network Isolation filters in accordance with a pre-programmed policy set by the Management Console
- Sending an alert to the Enterprise Management Framework
- Logging the event to the local event log.

Agent Presence Flow

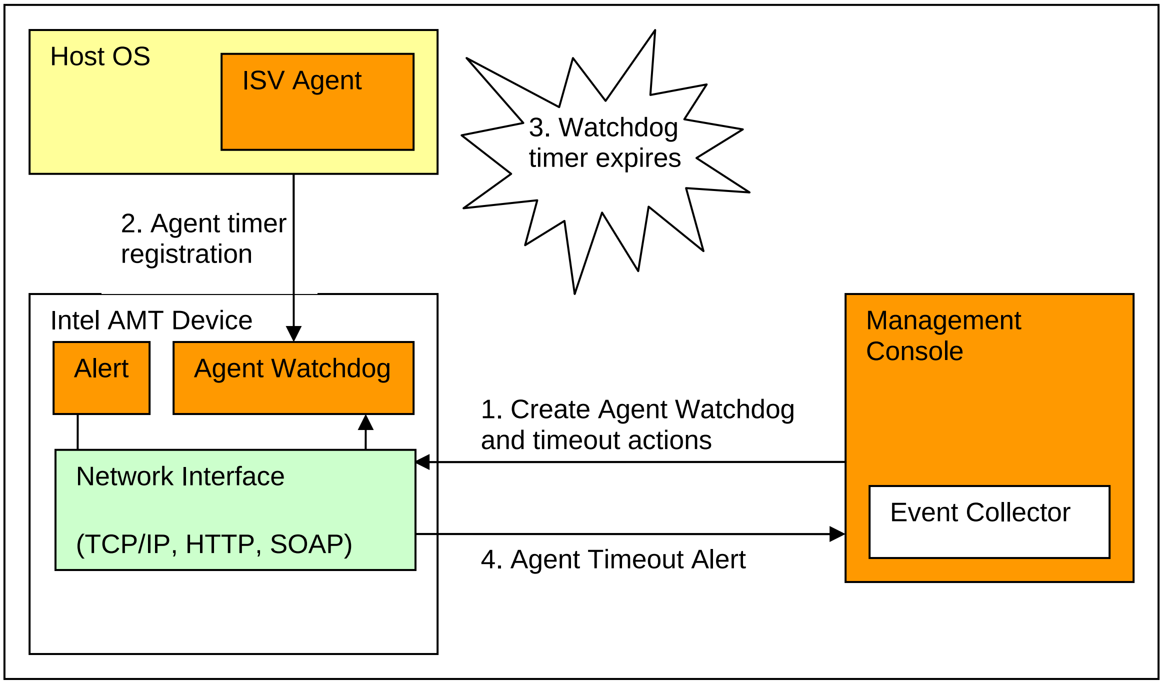


Figure 4: Agent Presence Usage Model

The components required to configure AP are:

- Management Console (MC) application (running on a system elsewhere on the network)
- Intel AMT device
- Software agent application running in the host OS running on the Intel AMT system platform

The MC application is used to configure the Intel AMT device with AP settings such as agent watchdog creation and timeout actions and any related required CB policies.

During initialization, the software agent registers with the local Intel AMT device. Once registered, the software agent periodically sends heartbeat signals to the Intel AMT device indicating it is still active.

If the Intel AMT device does not receive a heartbeat signal from the local software agent within the heartbeat interval timeout period, the AP actions are triggered.

Management Console Creates an Agent Watchdog

1. The MC application calls `ConsoleWatchdogCreate()` to create an Agent Presence (AP) watchdog for an agent. The function specifies:
   a. Agent ID that uniquely identifies the agent
   b. Agent description (optional)
   c. Maximum number of seconds between agent heartbeat calls

   d. Maximum number of seconds allowed for the agent to register after the OS is booted

2. Optionally the remote management application calls `ConsoleWatchdogSetCbPolicy`(), to define a CB policy to enable / disable when the agent state changes. See Circuit Breaker in AP.

3. The remote management application calls `ConsoleWatchdogSetActions`(), to specify a set of watchdog actions (a state transition table). Each action specifies what happens when the agent state changes, from a specific state, to a specific new state. For the list of possible states see Agent Presence States.

   a. The possible actions for a state change are:

Action	Description
`ActionEventOnTransition`	Specifies whether an Event should be created in the Intel AMT Event Manager when the application watchdog transitions from OldState to NewState.
`ActionCB`	A Circuit Breaker Action which may be applied when the application watchdog transitions from OldState to NewState. The action can be ActivateCBPolicy, DeactivateCBPolicy or null.

4. This creates an Agent Presence watchdog timer, associates a timeout event within the Intel AMT device and initiates the countdown of the timer.

5. When an agent state changes, the actions defined for this state change are executed by the Intel AMT device. State changes can occur as a result of:

   a. The agent registering with Intel AMT from the local host: using `AgentWatchdogRegister`()

   b. The agent sends a heartbeat: using `AgentWatchdogHeartbeat`()

   c. The timer expires: The agent has not registered or has not sent a heartbeat signal, so timeout expires.

   d. The agent reports shutdown: using `AgentWatchdogShutdown`().

Agent Presence States

The possible watchdog states are:

State	Description
`WatchdogStateNotStarted`	The agent has not registered with Intel AMT
`WatchdogStateRunning`	The agent has registered with Intel AMT and the associated timer is actively counting down
`WatchdogStateExpired`	The watchdog timer associated with this agent has counted down to a 0 value.
`WatchdogStateStopped`	The watchdog timer associated with this agent has been stopped via an API or network command
`WatchdogStateSuspended`	System has entered suspend (S1, S2, S3) state
`WatchdogStateAny`	Not a real agent state. Used in `ConsoleWatchdogSetActions( )` for specifying “don't care” for the old state of the agent.

Note: In suspended state (system is in S1, S2, S3) the Intel AMT timer countdown is suspended.

Circuit Breaker in AP

As explained in Active Policy, AP controls one entry in the **Enabled** CB policies list. To decide what should be the content of this entry, AP does the following, when an Agent state changes:

1. For each monitored agent, check the last state transition of the agent
2. For this last state change: If at least one agent specifies `ActionCB` = `ActivateCbPolicy`, enable the Agent Presence CB policy.
3. If no agents are in an active state, disable the Agent Presence CB Policy.

See CB Filters Policies.

AgentID

Each agent is identified by an AgentID GUID. The AgentID value is shared between the agent and the MC application. After the MC application creates an Agent Presence instance in the Intel AMT device, the software agent can register using AgentWatchdogRegister() specifying its AgentID. Upon registration, Intel AMT resets the associated watchdog timer. The actions associated with an agent state change can only be set by a MC application over a network connection.

Watchdog Timer

Intel AMT maintains a timer for each registered agent. The Intel AMT device is responsible for executing any actions associated with a change of state for the agent. Intel AMT will decrement these timers only when the system is in an S0 state. During other power states all timers are suspended.

CB and AP Usage Examples

Create a CB Filter

Create a RateLimit Filter

To create a rate limit filter for limiting the IP packets received from host 22.33.44.55 to 200 packets per second do the following:

```
CircuitBreakerFilterType filter;
filter.FilterName = "RateLimit <= 200";
filter.FilterDirection = FilterDirectionReceive;
filter.FilterProfile = FilterProfileRateLimit;
filter.FilterProfileData = 200;
filter.FilterPacket.PacketIP = CircuitBreakerPacketIPType;
filter.FilterPacket.PacketIP.IPPacket.IPv4 = CircuitBreakerIPv4Type;
filter.FilterPacket.PacketIP.IPPacket.IPv4.IPv4Desc->IPAddressDirection
= FilterDirectionSource;
filter.FilterPacket.PacketIP.IPPacket.IPv4.IPv4Desc->Address =
"22.33.44.55";
```

```
filter.FilterPacket.PacketIP.IPPacket.IPv4.IPv4Desc->AddressMask =
"255.255.255.255";
filter.FilterPacket.PacketIP.NextProtocol = NULL;
filter.ActionEventOnMatch = true;
uint32 handle;
CbFilterCreate(&filter, &handle);
```

Create a Drop Filter

To create a Drop filter for blocking TCP packets originating from IPv4 address 33.44.55.66 with a source port > 1023 do the following:

```
CircuitBreakerFilterType filter;
filter.FilterName = "33.44.55.66";
filter.FilterDirection = FilterDirectionSource;
filter.FilterProfile = FilterProfileDrop;
filter.FilterPacket.PacketTCP = CircuitBreakerPacketTCPType;
filter.FilterPacket.PacketTCP.IPPacket.IPv4 = CircuitBreakerIPv4Type;
filter.FilterPacket.PacketTCP.IPPacket.IPv4.IPv4Desc-
>IPAddressDirection = FilterDirectionReceive;
filter.FilterPacket.PacketTCP.IPPacket.IPv4.IPv4Desc->Address =
"33.44.55.66";
filter.FilterPacket.PacketTCP.IPPacket.IPv4.IPv4Desc->AddressMask =
"255.255.255.255";

IPLayeredPortType port;
port.IPLayeredPortRangeSource = CircuitBreakerIPLayeredPortRangeType
port.IPLayeredPortRangeSource.PortMin = 1024;
port.IPLayeredPortRangeSource.PortMax = 0xFFFF;

filter.FilterPacket.PacketTCP.IPLayeredPort = &port;
filter.FilterPacket.PacketTCP.TcpFlags = NULL;


filter.ActionEventOnMatch = true;
uint32 handle;
CbFilterCreate(&filter, &handle);
```

Create and Enable a CB Policy

To create a CB policy:

Call `CbQueryCapabilities ()` to get the `HardwareID`.

Call `CbFilterCreate ()` for each filter and save the `FilterHandle`.

Call `CbPolicyCreate ()` using all the saved `FilterHandles` and save the `PolicyCreationHandle.`

Call `CbPolicyEnable ()` using the `HardwareID` and `PolicyCreationHandle`.

Disable a CB Policy

To disable a CB policy that was enabled by a Management Console (MC) for a specific HardwareID:

Call CbPolicyDisable()using the HardwareID.

If no HardwareID is specified, this command will disable the MC CB policy of all of the interfaces.

Remove a CB Policy

To remove a CB policy:

Call CbPolicyEnumerate()

Examine the returned CircuitBreakerPolicy structures to identify the policy desired, and save the PolicyCreationHandle.

Call CbPolicyDelete() using the selected PolicyCreationHandle.

Create an AP Watchdog

To create an AP watchdog:

First create a CB policy. See Create a CB policy.

Call ConsoleWatchdogSetCbPolicy using the desired HardwareIDs and CB policies.

Call ConsoleWatchdogCreate()

Specify the state transition table in ConsoleWatchdogActions[].

Call ConsoleWatchdogSetActions() with ConsoleWatchdogActions.

Local Agent Registration and Heartbeat Signals

Call AgentWatchdogRegister () with the AgentID of the agent.

Store SessionSequenceNumber, and AgentHeartbeatTime.

Every AgentHeartbeatTime seconds:

- Call AgentWatchdogHeartbeat()
- Increment SessionSequenceNumber by 1

CB and AP Diagram

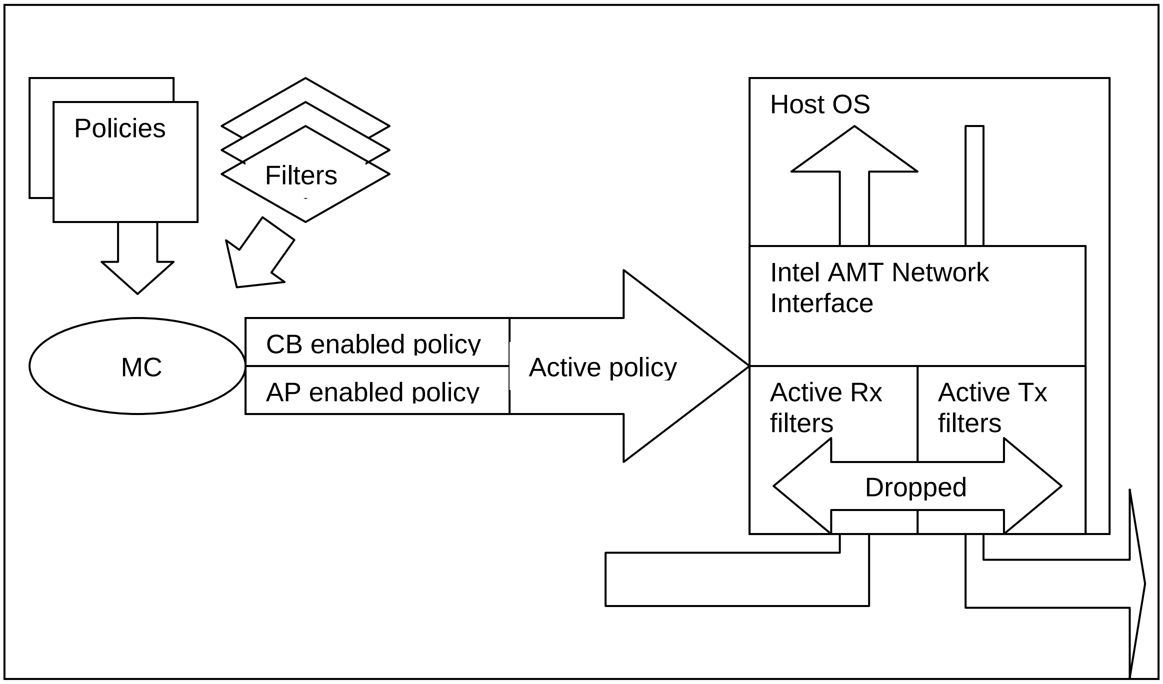


Figure 5: Agent Presence and Circuit Breaker

Figure 5 summarizes the AP and CB capabilities:

- There is a pool of defined filters.
- There is a pool of policies, each containing several filters.
- There are one or more AP watchdogs. They may enable a single policy.
- MC may enable a single policy.
- The enabled policy with the higher precedence is the active policy.
- The Tx filters in the active policy filter each transmitted packet.
- The Rx filters in the active policy filter each received packet.

Use Cases

The following four use cases demonstrate the capabilities of Circuit Breaker and Agent Presence. Code that implements the use cases is included in the Intel AMT SDK.

Use Case #1 - Host is Infected by a Virus

A system has been identified by a central management console as possibly infected with a worm and the central console would like to restrict the system so that it can communicate with only one subnet.

1. Management Console (MC) defines an INSPECTION AND REPAIR CB policy (priority 99). In this policy, network traffic is limited to the inspection and repair subnet (192.168.1.*).
2. MC defines several Rate Limit filters. For each filter, a PET is defined that will be sent to MC if the rate limit condition occurs:
   a. If the number of SYN packets sent from the host is greater than 1000 per second, Intel AMT sends a PET to MC.
   b. If the number of ICMP (ping) packets sent from the host is greater than 500 per second, the Intel AMT sends a PET to MC.
3. MC receives PET messages indicating SYN or ping attacks.
4. MC places this host in the inspection and repair subnet by applying the INSPECTION AND REPAIR CB policy.
5. MC opens a trouble ticket for an operator to inspect and repair this host.
6. A technician receives the trouble ticket, repairs the host, and marks the trouble ticket as completed.
7. MC is notified that trouble ticket is closed. MC deactivates and disables the INSPECTION AND REPAIR CB policy.

Use case #2 - Local Anti Virus Application Determines Antivirus Signature File is Obsolete

In this use case an Anti Virus agent determines that its signature file is out-of-date, and the host needs to be quarantined.

1. Management console (MC) defines a QUARANTINE CB policy in which the only network traffic allowed is traffic from the management console to the host using the TCP port assigned to the Anti Virus for updating the signature file. All other network traffic is blocked (dropped).
2. The MC creates an agent watchdog for the local agent that is configured to activate the QUARANTINE CB policy for any agent state change to "stopped".
3. The MC also specifies that AP should deactivate the QUARANTINE CB policy for any agent state change to "running".
4. During host operation, the local Anti-Virus agent determines that its policy signature is out-of-date.
5. The local agent signals Intel AMT it is going down, by calling `AgentWatchdogShutdown`().
6. Intel AMT automatically applies the QUARANTINE CB policy to the network interface.
7. The local agent begins remediation activities with the management console to update its signature file.
8. When the local agent completes its remediation activities, it registers again with Intel AMT Agent Presence and starts sending heartbeats.
9. Intel AMT AP detects that the agent state has changed to running and reopens the network by deactivating the QUARANTINE CB policy.

Use case #3 - Antivirus Agent Crashes

The following use case prepares for and responds to an anti virus (AV) application crash:

1. Management Console (MC) creates the following CB filters:

- A default filter blocking all Receive traffic to the host.
- A default filter blocking all Transmit traffic from the host.

2. MC creates BLOCKING CB POLICY which uses the above two filters and has a precedence set to 9.
3. MC registers AV by calling ConsoleWatchdogCreate().
4. MC calls ConsoleWatchdogSetActions() to specify the actions taken on each AV state change:
    a. If state changes from ANY STATE to RUNNING:
        i. Write event to event log.
        ii. Send PET to Management Console.
        iii. Disable the BLOCKING CB POLICY.
    b. If state changed from ANY STATE to NOT STARTED or from ANY STATE to EXPIRED, or from ANY STATE to STOPPED:
        i. Write event to event log.
        ii. Send PET to management console.
        iii. Enable BLOCKING POLICY.
5. The Host AV starts and registers to Intel AMT.
6. The AV state is monitored by Intel AMT. When the state changes from NOT STARTED to RUNNING, Intel AMT:
    a. Writes an event to event log.
    b. Sends a PET alert to the management console.
    c. Disables the BLOCKING POLICY.
7. AV locally registers to Intel AMT Agent Present (AP).
8. AV sends heartbeats to AP.
9. AV crashes.
10. The Intel AMT Agent Present timer expires. The AV state changes from RUNNING to EXPIRED. Intel AMT:
        i. Writes event to event log.
        ii. Sends PET to management console.
        iii. Enables BLOCKING POLICY.
11. The block policy isolates the Host from the network.
12. MC detects the PET alert indicating AP expiration. MC does the following:
    a. Re-Boots the host using Intel AMT Remote Control and IDE-R.
    b. Runs Virus “Clean” Tools Remotely.
    c. Re-Boots the host using Intel AMT Remote Control from local disk.
13. AV on the host starts and registers to Intel AMT, which:
        i. Writes event to event log.

        ii. Sends PET to management console.

        iii. Disables the BLOCKING POLICY.

14. MC detects the PET alert indicating AP Running.

Use case #4 - Antivirus Agent is Disabled

In this use case, Intel AMT detects and responds to a user disabling an anti virus (AV) application:

1. Management Console (MC) creates the following CB filters:

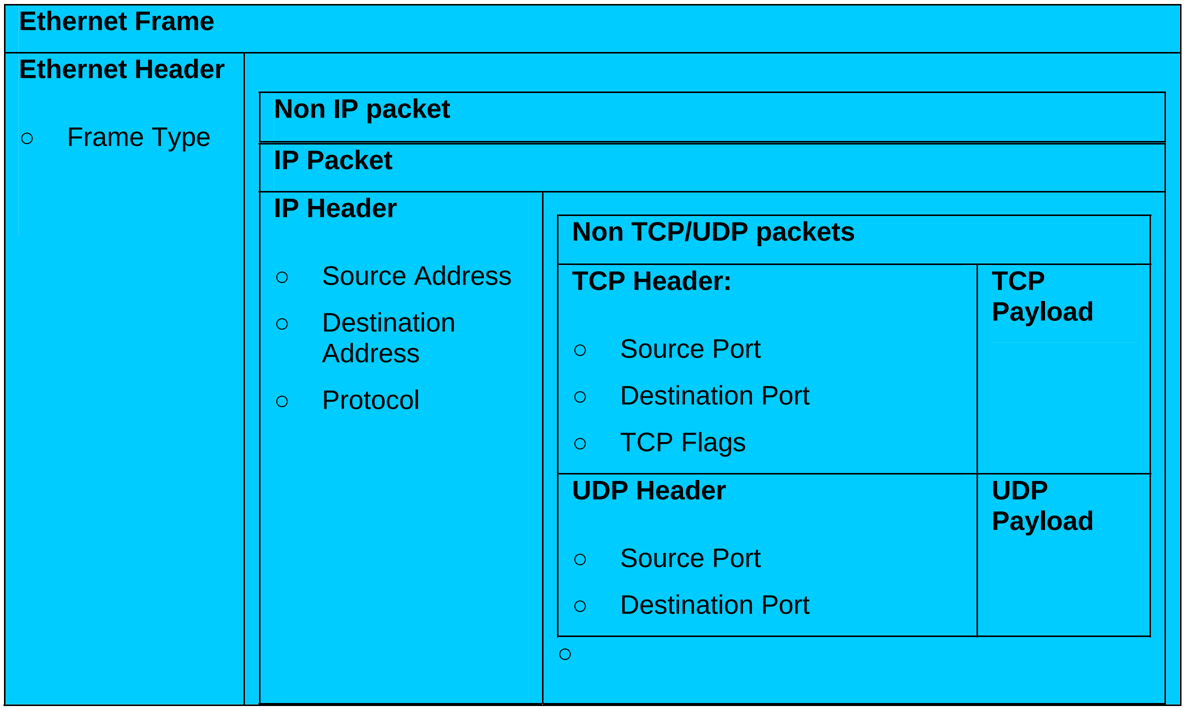
- A filter blocking all Receive traffic to the host.
- A filter blocking all Transmit traffic from the host.

2. MC creates a CB BLOCKING POLICY which uses the above 2 filters, with a precedence of 9.
3. MC registers AV application by calling ConsoleWatchdogCreate().
4. MC calls ConsoleWatchdogSetActions() to specify the actions taken in each AV state change:
    a. If the AV state changes from ANY STATE to RUNNING:
        i. Write event to event log.
        ii. Send PET to Management Console.
        iii. Disable the BLOCKING CB POLICY.
    b. If the state changes from ANY STATE to NOT STARTED or from ANY STATE to EXPIRED, or changed from ANY STATE to STOPPED:
        i. Write event to event log.
        ii. Send PET to MC.
        iii. Enable BLOCKING POLICY.
5. Host AV application starts and registers to Intel AMT AP by calling `AgentWatchdogRegister`().
6. Intel AMT monitors the AV application state.. When it changes from NOT STARTED to RUNNING, Intel AMT:
    a. Writes event to event log.
    b. Sends PET to MC.
    c. Disables the BLOCKING CB POLICY.
7. AV application sends heartbeats to AP.
8. AV application receives a user Disable request.
9. AV application tells Intel AMT it is going down, by calling `AgentWatchdogShutdown`().
10. Intel AMT Agent Present AV state changed from RUNNING to STOPPED. Intel AMT:
        i. Writes event to event log.
        ii. Sends PET to MC.
        iii. Enables BLOCKING POLICY.

11. The Host is now isolated from the network.
12. MC detects the PET indicating that the AP stopped. MC does the following:
    a. Establishes a Terminal Emulation session to the host.
    b. Enables the AV.
    c. Sends E-mail to user saying: Do not disable the AV application.
13. AV on the host starts and registers to Intel AMT.
14. Intel AMT:
    i. Writes event to event log.
    ii. Sends PET to MC.
    iii. Disables the BLOCKING CB POLICY.
15. MC detects the PET indicating AP Running.

Appendix A - Networking Packet Structures

Figure 5: Packet Structure illustrates the layout of Ethernet packets.



Ethernet Frame

Ethernet Header

- ○ Frame Type

Non IP packet

IP Packet

IP Header

- ○ Source Address
- ○ Destination Address
- ○ Protocol

Non TCP/UDP packets

TCP Header:

- ○ Source Port
- ○ Destination Port
- ○ TCP Flags

TCP Payload

UDP Header

- ○ Source Port
- ○ Destination Port

UDP Payload

○

Figure 5: Packet Structure

The layout shown in this appendix identifies the fields in a packet that a CB filter can evaluate. An Ethernet packet can contain:

- A non IP Packet
- An IP Packet, which can contain:
  - A TCP Packet
  - A UDP Packet
  - A Non TCP/UDP packet

Use the type `CircuitBreakerFilterType.FilterPacket` to specify the following packet header fields, for either transmit or receive packet streams.

A.1 Ethernet Header

- Frame Type (2 bytes)

A.2 IPv4 Header

- Source Address (4 bytes): use a mask to perform wild card IP address matches
- Destination Address (4 bytes): use a mask to perform wild card IP address matches
- Protocol Type – (1 byte) – see IANA protocol numbers

A.3 IPv6 Header

- Source Address (16 bytes): use a mask to perform wild card IP address matches
- Destination Address (16 bytes): use a mask to perform wild card IP address matches
- Protocol Type (1 byte) – see IANA protocol numbers

A.4 TCP Header

- Source Port (2 bytes): supports a range of port values
- Destination Port (2 bytes): supports a range of port values
- TCP flags: (6 bits in the 14th byte) most valuable with a rate limit filter – the SYN bit is the fifth bit

A.5 UDP Header

- Source Port (2 bytes): supports a range of port values
- Destination Port (2 bytes): supports a range of port values