



(19) **United States**

(12) **Patent Application Publication**  
**Hamilton et al.**

(10) **Pub. No.: US 2003/0177283 A1**

(43) **Pub. Date: Sep. 18, 2003**

(54) **APPLICATION PROGRAM INTERFACE**

(52) **U.S. Cl. .... 709/328**

(76) Inventors: **Thomas E. Hamilton**, Marlborough,  
MA (US); **Clifford S. Atwood**,  
Harvard, MA (US)

(57) **ABSTRACT**

Correspondence Address:  
**FISH & RICHARDSON PC**  
**225 FRANKLIN ST**  
**BOSTON, MA 02110 (US)**

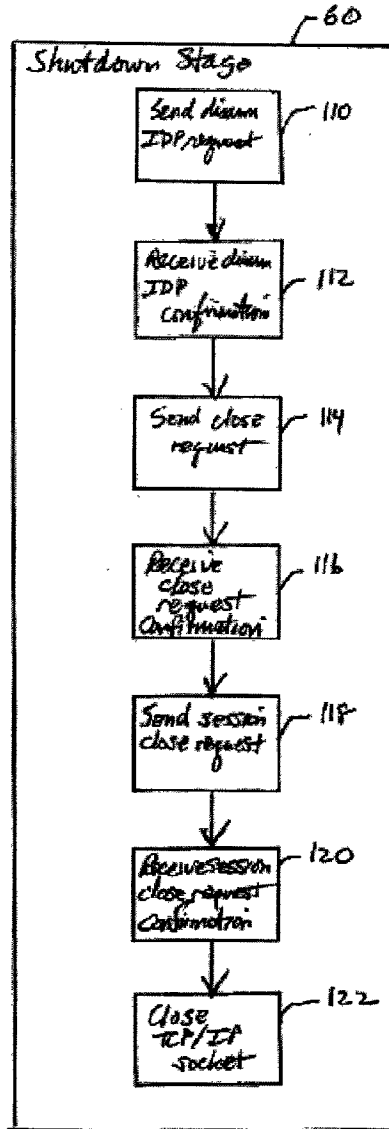
A method in a network is provided. The method includes receiving messages from an application program in an application program interface (API), and passing the messages from the API to a control process in a mobile service switching platform (MSSP). A system is also provided. The system includes a Gateway General Packet Radio Service Support Node (GGSN) linked to control process in a Mobile Service Switching Platform (MSSP), a group of globally connected computers linked to the control process, an application program interface (API) connected to the control process, and an application system executing an application process linked to the API.

(21) Appl. No.: **10/100,468**

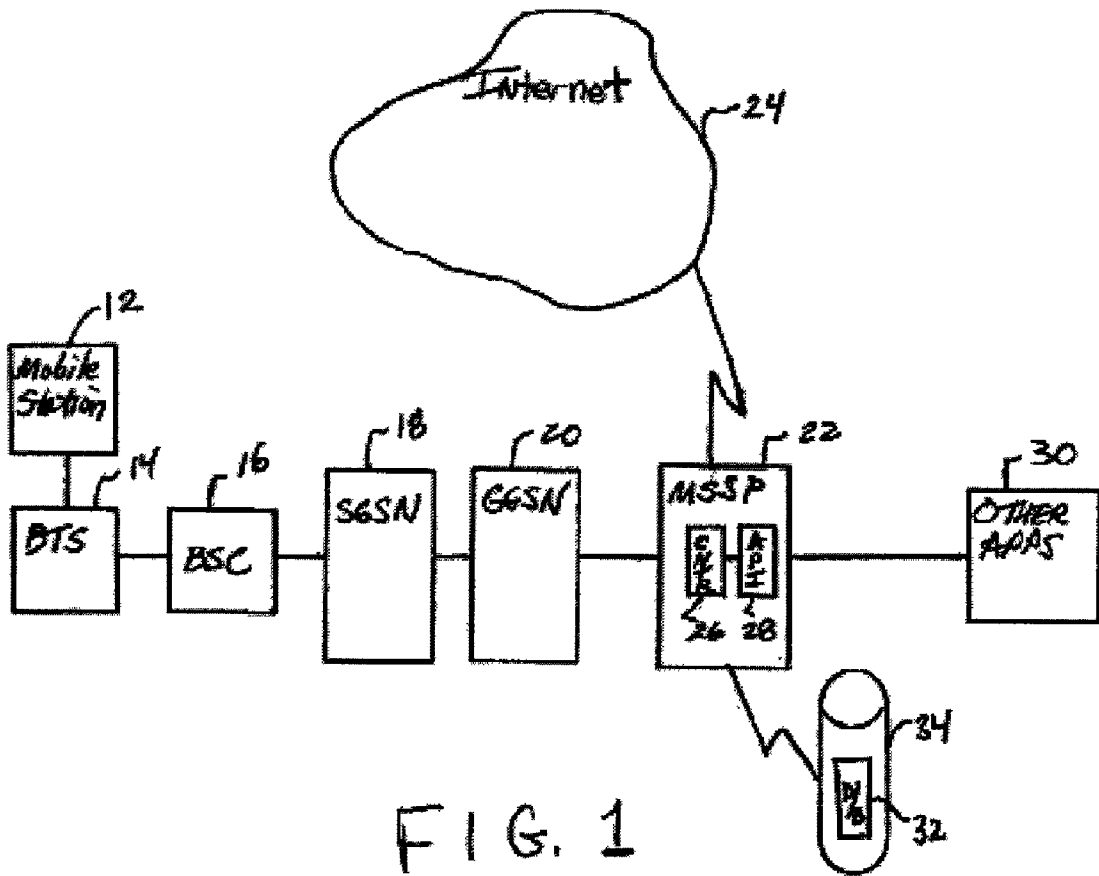
(22) Filed: **Mar. 18, 2002**

**Publication Classification**

(51) **Int. Cl.<sup>7</sup> ..... G06F 9/00; G06F 9/46**



10



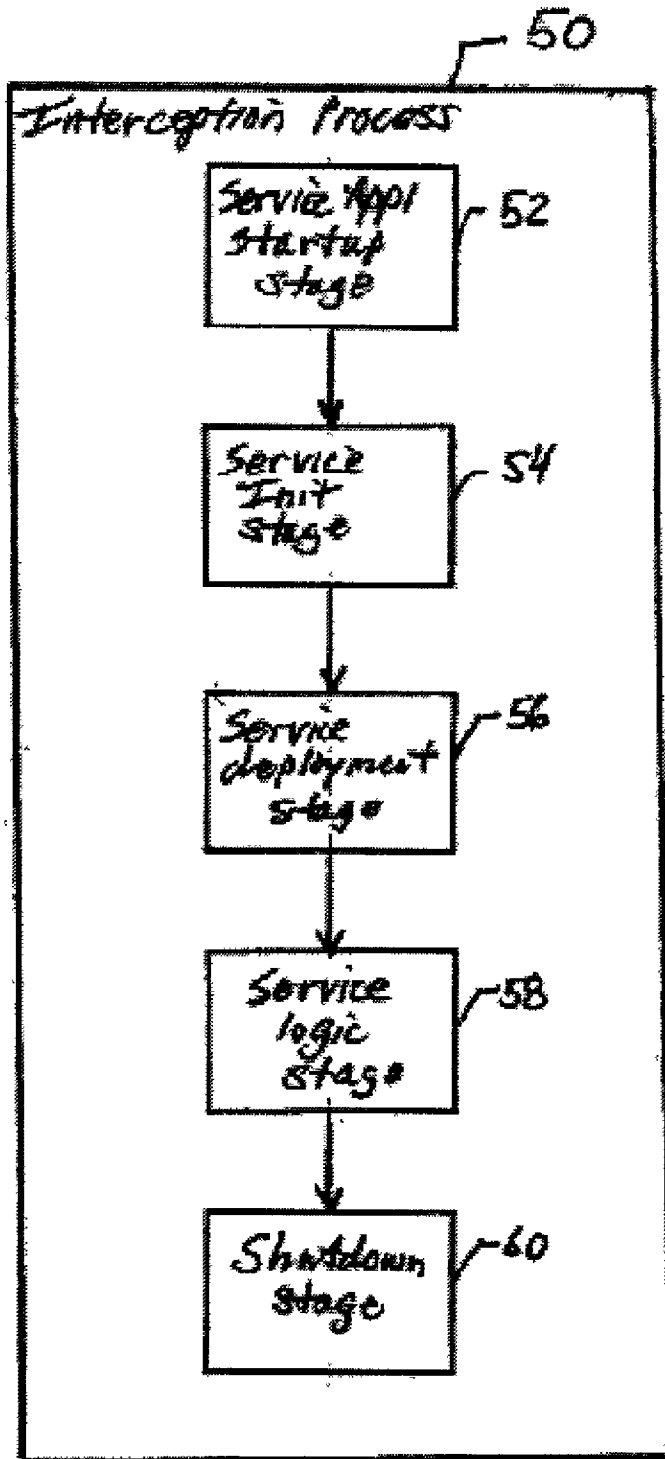


FIG. 2

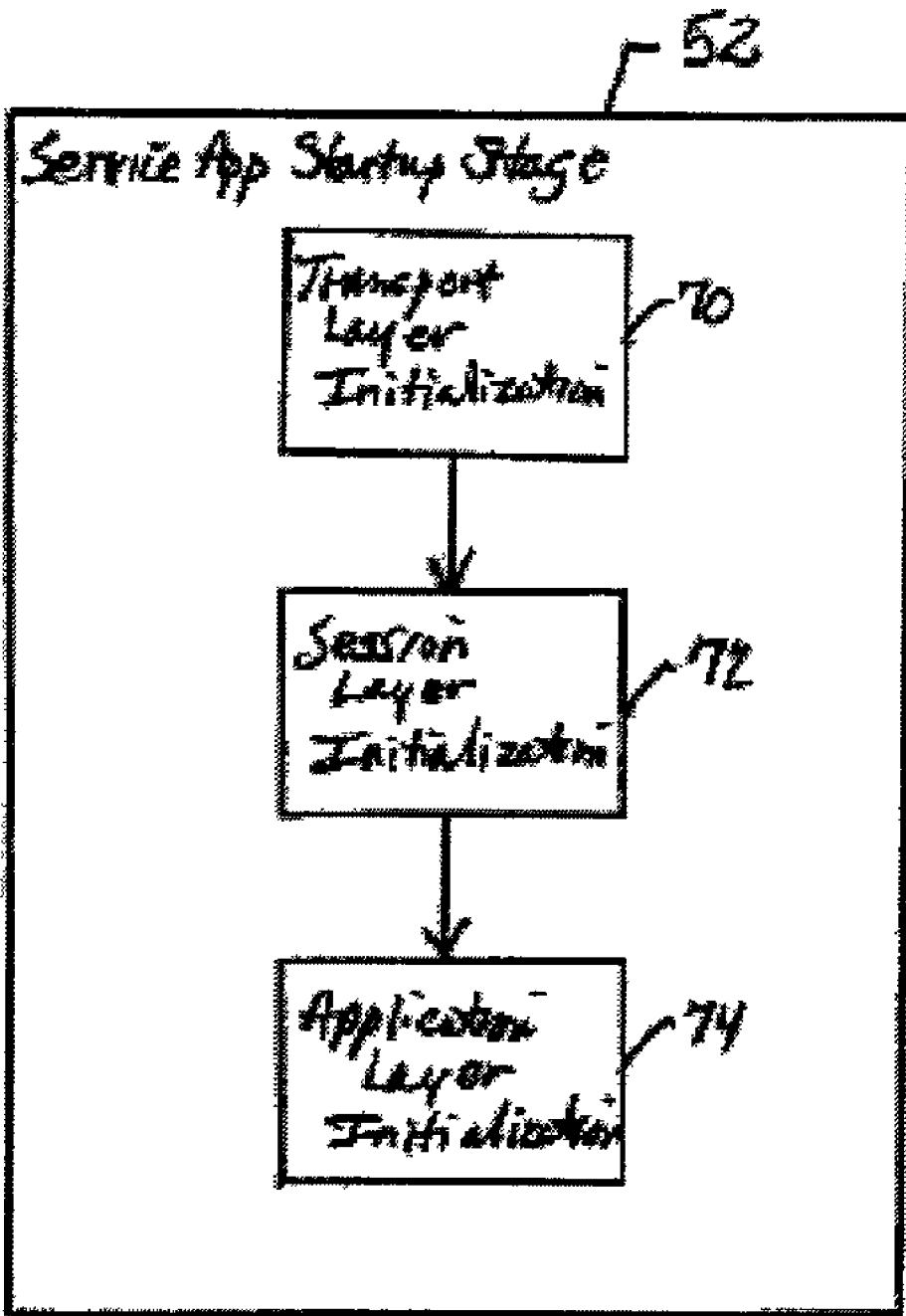


FIG. 3

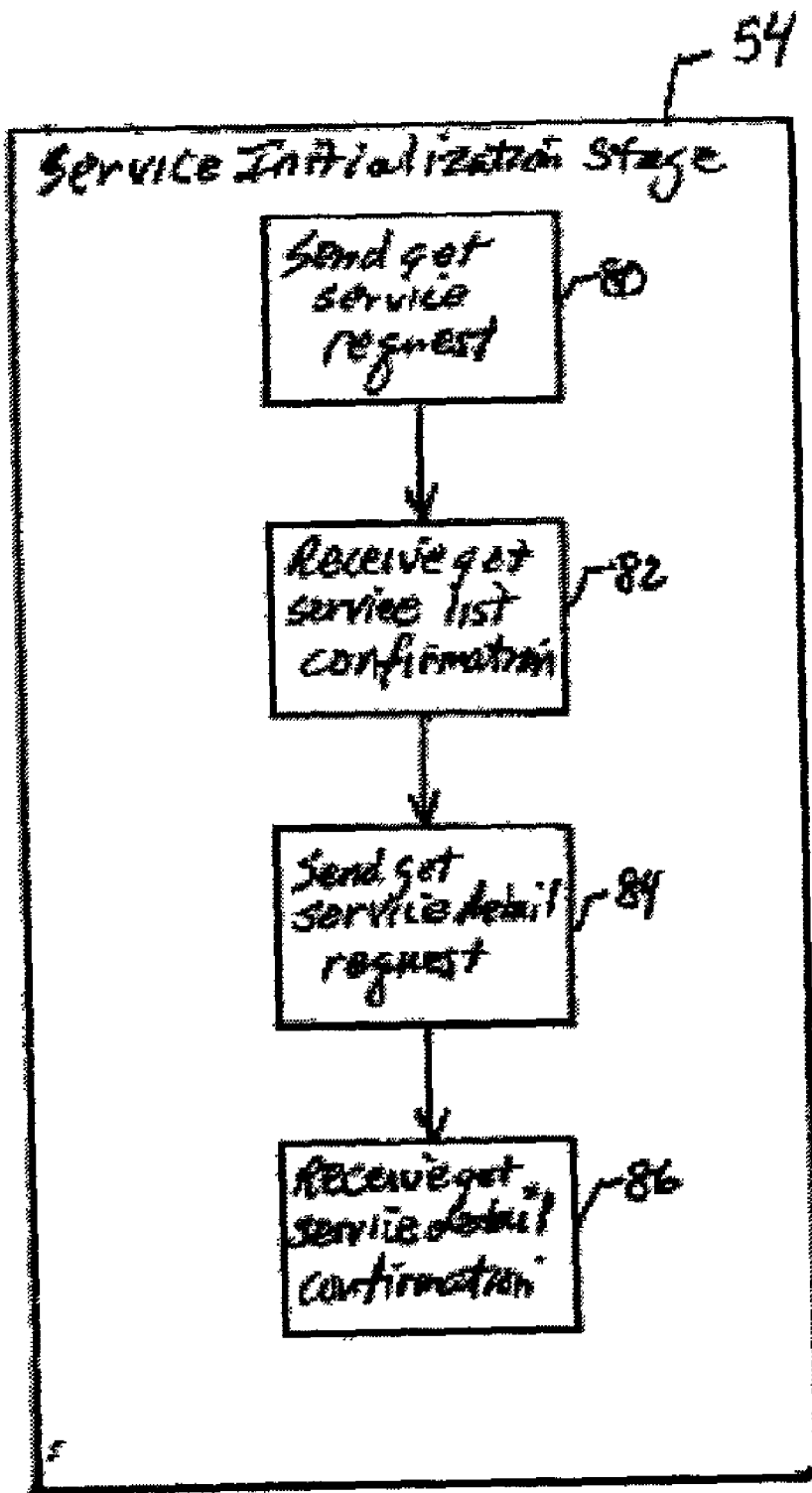


FIG. 4

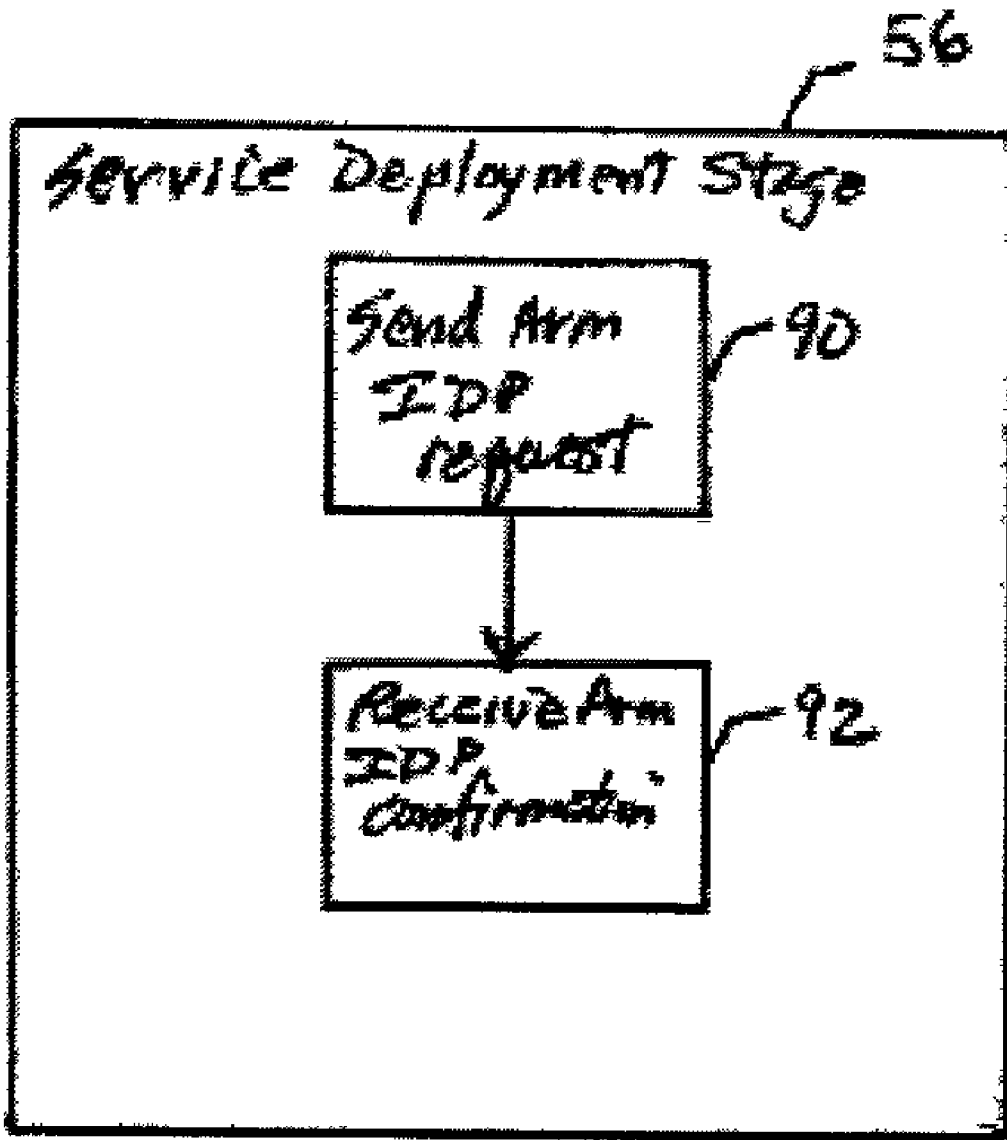


FIG. 5

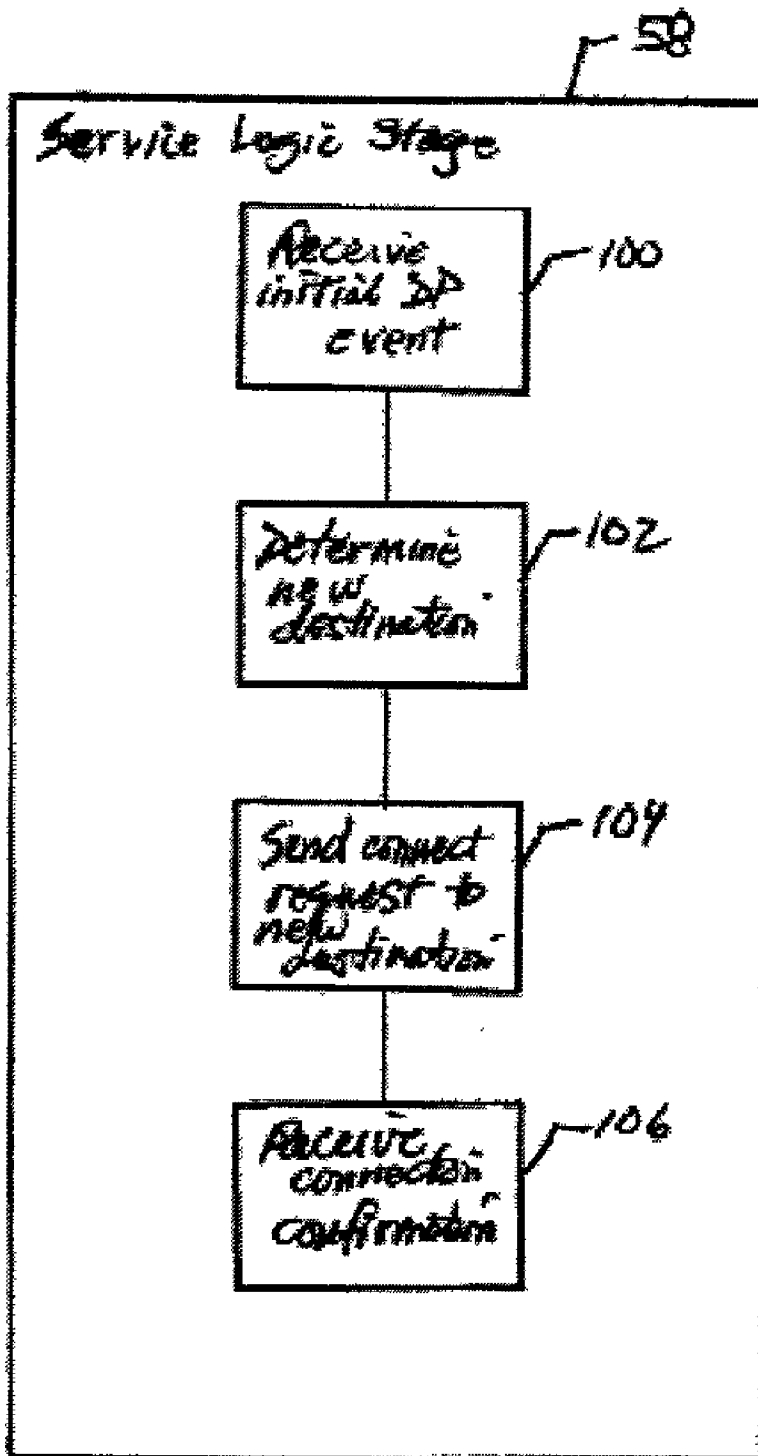


FIG. 6

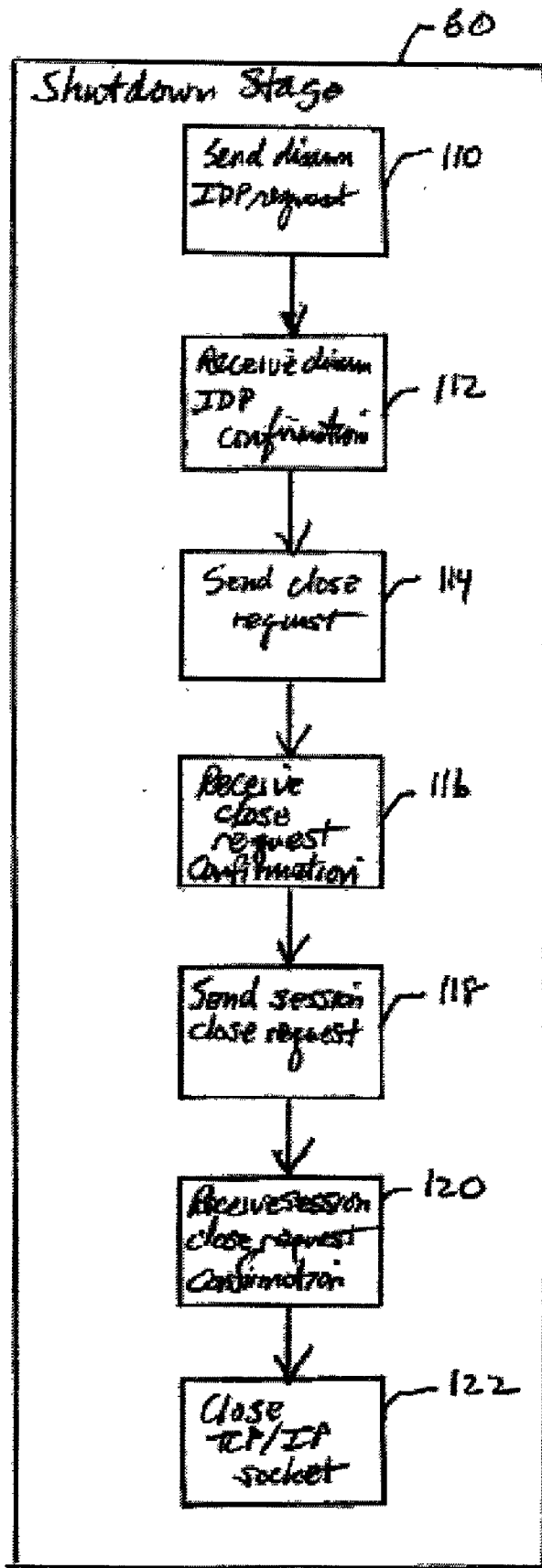


FIG. 7

130

DATA TYPE NAME	DEFINITION	BYTESIZE
BYTE	8-bit unsigned integer, range 0 to 255.	1
BYTE[n]	A fixed-size array of <i>n</i> consecutive 8-bit unsigned integer values (BYTES).	N
CHAR	8-bit signed integer, range -128 to 127	1
USHORT	16-bit unsigned integer, range 0 to 65,535 (0x0 ... 0xFFFF)	2
UINT	32-bit unsigned integer, range 0 to 4,294,967,295 (0x0 ... 0xFFFF FFFF)	4
ULONG	64-bit unsigned integer, range 0 to 18,446,744,073,709,551,615 (0x0 ... 0xFFFF FFFF FFFF FFFF)	8
CHAR[n]	A fixed length sequence of exactly <i>n</i> CHARs containing a UTF-8 character string. The string contained in this field may be of shorter length, indicated by a NULL (0x00) terminating character.	<i>n</i>
VARCHAR	A variable-length UTF-8 character string. Encoded as a USHORT giving the length of the string (including a NULL terminating character) followed by <i>length</i> characters. Padded as needed to a mod 4 boundary.	2 + N + alignment padding.
VARBYTE	A variable-length sequence of BYTE data. Encoded as a USHORT giving the length of the data followed by <i>length</i> bytes. Padded as needed to a mod 4 boundary.	2 + N + alignment padding.
TIME	A date and time value expressed as the number of seconds since midnight, January 1, 1970 Coordinated Universal Time (UTC).	4

FIG. 8

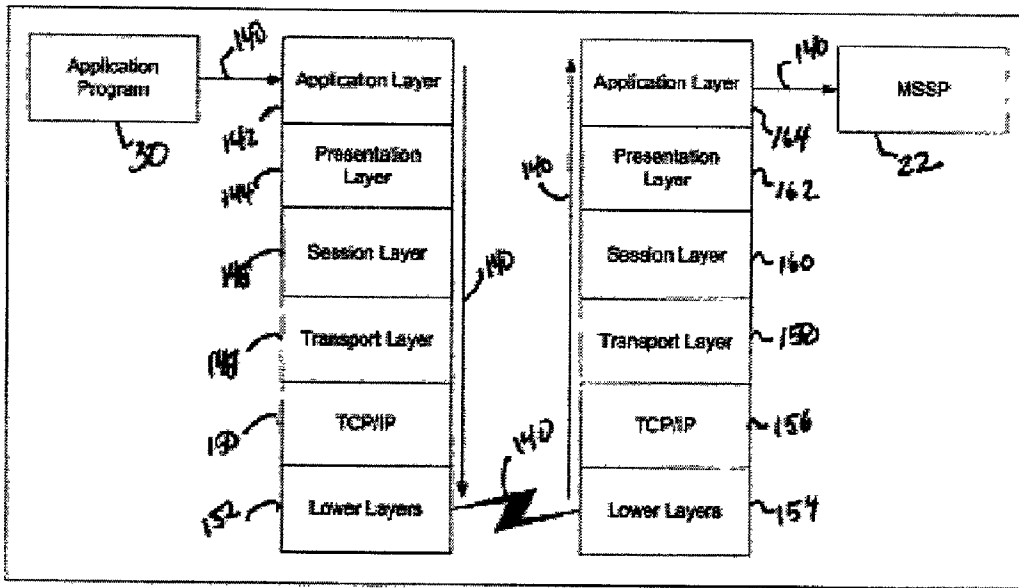


FIG. 9

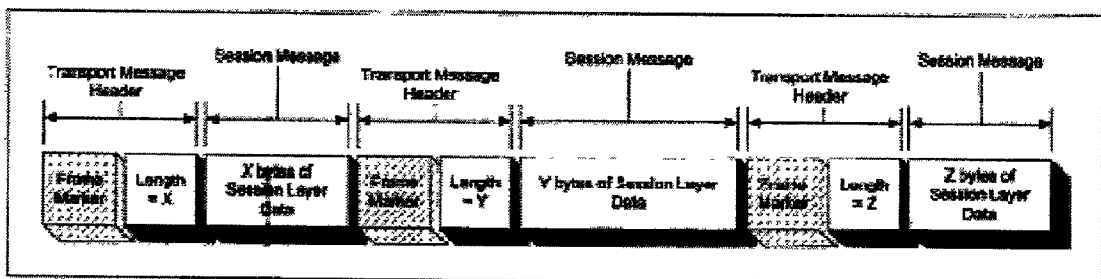


FIG. 10

ERRORCODE	VALUE	DESCRIPTION
MSSP_E_NONE	0	No error.
MSSP_E_INVALID_CONFIG_MASK	1	The given ConfigurationMask is invalid.
MSSP_E_INVALID_SVC_CAP_MASK	2	The given ServiceCapabilities mask is invalid.
MSSP_E_INVALID_API_METER_MASK	3	The given APIMeter mask is invalid.
MSSP_E_INVALID_FLOW_METER_MASK	4	The given FlowMeter mask is invalid.
MSSP_E_INVALID_VERSION	5	No protocol versions in the given range are supported.
MSSP_E_INVALID_OPERATION_CODE	6	The given OperationCode is invalid.
MSSP_E_INSUFFICIENT_RESOURCES	7	Insufficient resources to process request (out of memory, etc.)
MSSP_E_INVALID_OPERATOR_ID	8	The given OperatorID is not valid.
MSSP_E_INVALID_SESSION_GROUP_ID	9	The given SessionGroupID is not valid.
MSSP_E_OBJECT_IN_USE	10	The request cannot operation cannot be performed while the object is in use.
MSSP_E_INVALID_SESSION_ID	11	The given SessionID is not valid.
MSSP_E_INVALID_IP_ADDRESS_TYPE	12	An invalid IPAddressType was given.
MSSP_E_INVALID_DP_CLASS	13	The given DetectionPointClass is not valid.
MSSP_E_INVALID_DP	14	The given DetectionPoint is not valid.
MSSP_E_INVALID_ATTRIBUTE_MASK	15	The given AttributeMask is invalid.
MSSP_E_INVALID_CRITERIA_MASK	15	The given CriteriaMask is invalid.
MSSP_E_TAKE_CONTROL_INVALID	17	The given TakeControl value is invalid.
MSSP_E_NOT_A_TRIGGER	18	The specified detection point cannot be armed as a trigger.
MSSP_E_INVALID_RESOURCE_ID	19	The given ResourceID is not valid.
MSSP_E_INVALID_FLOW_ID	20	The given FlowID is not valid.
MSSP_E_INVALID_CONTROL_ID	21	The given ControlID is not valid.
MSSP_E_INVALID_STATE	22	Trigger response received when state machine not suspended at the trigger.
MSSP_E_INVALID_CONTROL_FLAG	23	An invalid ControlFlag value was given.
MSSP_E_INVALID_EVENT_MASK	24	An invalid EventReportMask value was given.
MSSP_E_INVALID_TRIGGER_MASK	25	An invalid TriggerMask value was given.
MSSP_E_INVALID_CONTROL_TAG	26	An invalid ControlTag value was given.
MSSP_E_INVALID_ACTION_CODE	27	An invalid ActionCode value was given.
MSSP_E_INVALID_METER_CLASS	28	An invalid MeterClass value was given.
MSSP_E_STL_DOWN	29	STL entity down.
MSSP_E_NOT_AUTHORIZED	30	The application is not authorized to use the requested feature.
MSSP_E_INVALID_STATS_TYPE	31	An invalid StatsType value was given.
MSSP_E_INVALID_INTERVAL	32	An invalid Interval value was given.
MSSP_E_NOT_AN_IDP	33	The specified detection point cannot be armed as an IDP.
MSSP_E_INVALID_FILTER		The AccessFilter indexed by ErrorArg is invalid.

FIG. 11

180

Feature Category	Description
182 Common Services	Messages common to all applications and available regardless of application privileges.
184 Initial Detection Point	Messages that allow an application to arm and disarm Initial Detection Points (IDPs) and to service IDP events.
186 Event Reporting	Messages that allow an application to request additional event reports after an IDP event.
188 Service Filter	Messages that allow an application to specify programmed behavior at an initial detection point without requiring the involvement of the application each time.
190 Meter Configuration	Messages that allow an application to configure the data elements that will be metered by the MSSP.
192 Charge Notification	Messages that allow an application to request byte based reporting.
194 Charge Plan	Messages that allow an application to indicate the cost of the services provided and record the charge plan used in the MSSP detail record.
196 Detail Record Control	Messages that allow an application to control when MSSP detail records are written.
198 Statistics	Messages that allow an application to obtain various statistics collected by the MSSP.
200 Application Monitor	Messages that allow an application to monitor the status of other applications.

FIG. 12

## APPLICATION PROGRAM INTERFACE

### TECHNICAL FIELD

[0001] This invention relates to an application program interface (API).

### BACKGROUND

[0002] In general, an application program interface (API) is a specific method prescribed by a computer operating system or by another application program by which a programmer writing an application program can make requests of the operating system or another application. More specifically, an API is a formalized set of software calls and routines that can be referenced by an application program in order to access supporting services.

### SUMMARY

[0003] In aspect, the invention features a method including, in a network, receiving messages from an application program in an application program interface (API), and passing the messages from the API to a control process in a mobile service switching platform (MSSP).

[0004] Embodiments may include one or more of the following. The network may be a wireless network. The wireless network may be a second generation wireless network, a GSM network, a GPRS-enabled GSM network or a TDMA network. The wireless network may be a CDMA network, a UMTS network, a TETRA network or a Tetrapol network. The wireless network may be a DECT network, an AMPS network, a WLAN or a third generation wireless network.

[0005] The API may provide a protocol that allows the application program to control switching and routing functions in the MSSP.

[0006] The API may provide a protocol that allows the application program to redirect packet flow through the MSSP on a per-flow basis.

[0007] The API may provide a protocol that allows the application program to control policy decisions within the MSSP.

[0008] The API may include a protocol that allows the application program to arm initial detection points (IDPs) and services associated IDP events in the control process.

[0009] The API may include a protocol that allows the application program to disarm IDPs and service associated ICP events in the control process.

[0010] The API may include a protocol that allows the application program to request event reports.

[0011] The API may include a protocol that allows the application program to specify programmed behavior at a detection point in the control process.

[0012] The API may include a protocol that allows the application program to configure data elements that are metered by the control process of the MSSP.

[0013] The API may include a protocol that allows the application program to request byte-based reporting. The reporting may be session-based or flow-based.

[0014] The API may include a protocol that allows the application program to specify a cost of services provided.

[0015] The API may include a protocol that allows the application program to record a charge plan used in a detail record and a protocol that allows the application program to control when the detail record is written.

[0016] The API may include a protocol that allows the application program to obtain statistics for a session managed by the application program.

[0017] The API may include a protocol that allows the application program to obtain statistics for a flow managed by the application program.

[0018] The API may include a protocol that allows the application program to monitor a status of other applications connected to the control process of the MSSP.

[0019] In another aspect, the invention features an application program interface (API) including a set of application layer protocols that allows exchange of messages between an application process and a control process of a Mobile Service Switching Platform (MSSP) using Transmission Control Protocol/Internet Protocol (TCP/IP) network services.

[0020] In embodiments the set of application layers protocols may include a protocol that allows the application process to arm initial detection points (IDPs) and services associated IDP events in the control process.

[0021] The set of application layers protocols may include a protocol that allows the application process to disarm initial detection points (IDPs) and services associated IDP events in the control process.

[0022] The set of application layers protocols may include a protocol that allows the application process to request event reports from the control process.

[0023] The set of application layers protocols may include a protocol that allows the application process to specify programmed behavior at a detection point in the control process.

[0024] The set of application layers protocols may include a protocol that allows the application process to configure data elements that are metered by the control process.

[0025] The set of application layers protocols may include a protocol that allows the application process to request byte-based reporting in the control process. The reporting may include session-based reporting or flow-based reporting.

[0026] The set of application layers protocols may include a protocol that allows the application process to specify a cost of services provided by the MSSP.

[0027] The set of application layers protocols may include a protocol that allows the application process to record a charge plan used in a detail record stored in the MSSP.

[0028] The set of application layers protocols may include a protocol that allows the application process to control when the detail record is written.

[0029] The set of application layers protocols may include a protocol that allows the application process to obtain statistics for a session managed by the application process.

[0030] The set of application layers protocols may include a protocol that allows the application process to obtain statistics for a flow managed by the application process.

[0031] The set of application layers protocols may include a protocol that allows the application process to monitor a status of other application processes connected to the control process.

[0032] In another aspect, the invention features a system including a Gateway General Packet Radio Service Support Node (GGSN) linked to control process in a Mobile Service Switching Platform (MSSP), a group of globally connected computers linked to the control process, an application program interface (API) connected to the control process, and an application system executing an application process linked to the API.

[0033] In embodiments, the system may include a General Packet Radio Service Support Node linked to the GGSN. The system may include a Base Station Controller (BSC) linked to the General Packet Radio Service Support Node. The system may include a Base Transceiver Station (BTS) linked to the BSC and a mobile station (MS) linked to the BTS.

[0034] The API may include a set of application layer protocols that allows exchange of messages between the application process and the control process.

[0035] The set of application layers protocols may include a protocol that allows the application process to arm initial detection points (IDPs) and services associated IDP events in the control process.

[0036] The set of application layers protocols may include a protocol that allows the application process to disarm initial detection points (IDPs) and services associated IDP events in the control process.

[0037] The set of application layers protocols may include a protocol that allows the application process to request event reports from the control process.

[0038] The set of application layers protocols may include a protocol that allows the application process to specify programmed behavior at a detection point in the control process.

[0039] The set of application layers protocols may include a protocol that allows the application process to configure data elements that are metered by the control process.

[0040] The set of application layers protocols may include a protocol that allows the application process to request byte-based reporting in the control process. Reporting may be session-based or flow-based.

[0041] The set of application layers protocols may include a protocol that allows the application process to specify a cost of services provided by the MSSP.

[0042] The set of application layers protocols may include a protocol that allows the application process to record a charge plan used in a detail record stored in the MSSP.

[0043] The set of application layers protocols may include a protocol that allows the application process to control when the detail record is written.

[0044] The set of application layers protocols may include a protocol that allows the application process to obtain statistics for a session managed by the application process.

[0045] The set of application layers protocols may include a protocol that allows the application process to obtain statistics for a flow managed by the application process.

[0046] The set of application layers protocols may include a protocol that allows the application process to monitor a status of other application processes connected to the control process.

[0047] Embodiments of the invention may have one or more of the following advantages.

[0048] The Application Program Interface (API) provides an application layer protocol that exchanges messages with a Mobile Service Switching Platform (MSSP) using simple TCP/IP network services that are available on almost all computer platforms.

[0049] The API provides a set of protocols that allow service logic contained in an external application program to control switching/routing functions of a Mobile Service Switching Platform.

[0050] The API provides a protocol for an operator to limit the scope of an application's detection points, in which a detection point is a defined place in a state machine of a control entity where application event reporting and/or control is possible.

[0051] The API provides a protocol that is common to all applications, regardless of application privileges.

[0052] The API provides a protocol that allows an application to arm and disarm Initial Detection Points (IDPs) in a Mobile Service Switching Platform (MSSP) and service associated IDP events, where an IDP is defined as a detection point armed so as to create a new control dialog with an application when conditions match given criteria.

[0053] The API provides a protocol that allows an application to request additional event reports subsequent to an Initial Detection Point event. When the IDP that initiates a control dialog is a trigger, the application typically requests additional event reports.

[0054] The API provides a protocol that allows an application to specify programmed behavior at a Detection Point (DP) that does not require the involvement of the application. Messages are used to match incoming requests to determine if the predefined service interaction should be executed. The matching process is similar to the process used for Initial Detection Points in general and wildcards may be used in the fields to be matched. If a flow matches the criteria, the actions specified in the remainder of the message will be carried out with no application involvement. Actions that may be specified include the reporting of events as well as redirecting a request to a specified redirection address and port number. A message is used to determine which events will be reported in the future for the flow if event reports are required. Criteria to be matched may not overlap with armed Initial Detection Point criteria. If the request cannot be completed for any reason a message will be returned with a matching RequestID and an error code indicating the nature of the failure. If the request

completes successfully, another message is returned. Service Filtering remains active until cancelled by a specific message request.

[0055] The API provides a protocol that allows an application to configure data elements that are metered by a Mobile Service Switching Platform (MSSP).

[0056] The API provides a protocol that allows an application to request byte based reporting. Reporting may be requested on a session or flow basis. Session based charge notification effectively causes the same charge notification criteria to be applied to all flows in the session. Registering for charge notification events causes the number of bytes of the specified type transferred in uplink and downlink directions to be metered. Each time a reporting threshold is reached a message is sent from the MSSP to the application indicating the number of bytes that have been transferred, and counters are reset and begin counting towards the threshold again. Charge notification continues until the flow terminates or charge notification is explicitly cancelled by a cancel request. A packet is the atomic unit counted and each packet either falls before the count is evaluated or after the count is evaluated. As a result, charge notification may not occur exactly on the byte count specified. For example, if notification was requested every 10K bytes, the notification may occur at 10.5 Kbytes if the packet that brought the count over 10K was slightly greater than 500 bytes. The actual counter values are provided in a message.

[0057] The API provides a protocol that allows an application to indicate a cost of the services provided and record a charge plan used in an MSSP detail record.

[0058] The API provides a protocol that allows an application to control when MSSP detail records are written.

[0059] The API provides a protocol that allows an application to obtain various statistics for a session or flow managed by the application.

[0060] The API provides a protocol that allows an application to monitor the status of other applications connected to the same MSSP instance.

[0061] The API provides a protocol that allows the redirection of packet flow on a per-flow basis.

[0062] Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

#### DESCRIPTION OF DRAWINGS

[0063] FIG. 1 is a block diagram of a network.

[0064] FIG. 2 is a flow diagram of an interception process.

[0065] FIG. 3 is a flow diagram of the service application startup stage of FIG. 2.

[0066] FIG. 4 is a flow diagram of the service initialization stage of FIG. 2.

[0067] FIG. 5 is a flow diagram of the service deployment stage of FIG. 2.

[0068] FIG. 6 is a flow diagram of the service logic stage of FIG. 2.

[0069] FIG. 7 is a flow diagram of the shutdown stage of FIG. 2.

[0070] FIG. 8 is a table of data types used by the API of FIG. 1.

[0071] FIG. 9 is a block diagram of a communication path.

[0072] FIG. 10 is a block diagram of a TCP/IP byte stream divided into session messages by the transport layer.

[0073] FIG. 11 shows a table listing sample error codes.

[0074] FIG. 12 shows a table listing sample feature categories.

#### DETAILED DESCRIPTION

[0075] Referring to FIG. 1, a network 10 is shown. The network 10, for example, may be a wireless network. The wireless network may be, for example, a second generation wireless network, a Global System for Mobile Communications (GSM) network, or a General Packet Radio System (GPRS) enabled GSM. The wireless network may be a Time Division Multiple Access (TDMA) network, a Code Division Multiple Access (CDMA) network, or a Universal Mobile Telecommunications System (UMTS) network. The wireless network may be a TETRA network, a Tetrapol network, a DECT network, an AMPS network, a wireless local area network (WLAN) or a third generation wireless network. By way of example, a GPRS enabled GSM network is described.

[0076] The network 10 includes a Mobile Station (MS) 12 connected to a Base Transceiver Station (BTS) 14. The BTS 14 is connected to a Base Station Controller (BSC) 16. In mobile communications, the MS 12 is a station located within a mobile service intended to be used while in motion or during halts at unspecified points. An example mobile station is a hand held cellular telephone.

[0077] The BTS 14 holds radio transceivers that define a cell and coordinates radio-link protocols with the MS 12. The BTS 14 is a component of the network 10 from which all signals are sent and received. The BTS 14, often called a cell phone tower, is linked to, and controlled by, a Base Station Controller (BSC) 16. The BSC 16 is a component in the network 10 that manages radio resources for one or more base transceiver stations, such as BTS 14, for example.

[0078] The BSC 16 is linked to a SGSN 18. The SGSN 18 is a General Packet Radio Service Support (GPRS) Node that serves GPRS mobile by sending or receiving packets via the BSC 16. The SGSN 18 is linked to a Gateway GPRS Support Node (GGSN) 20. The GGSN 20 acts as a gateway between a General Packet Radio Service (GPRS) network and a Packet Switched Public Data Network (PSPDN).

[0079] The GGSN 20 is linked to a Mobile Service Switching Platform (MSSP) server 22. The MSSP server 22 resides between the GGSN 20 and a globally networked group of computers, such as Internet 24. The MSSP server 22 analyzes all of the Internet Protocol (IP) data packets exchanged between the MS 12 and the Internet 24. A MSSP control process 26 provides the capability to set triggers or event notifications and increment counters based on IP flow characteristics. An IP flow can be thought of as an abstraction representing a movement of data between two endpoints, such as MS 12 and a server (not shown) residing on the Internet 24. The MSSP control process 26 uses these capabilities to implement internal services and detail report-

ing. An Application Program Interface (API) **28** links the MSSP control process **26** to external applications **30**. The API **28** provides a mechanism for the external applications **30** to control the MSSP control process **26** to provide intelligent services. The API **28**, in various embodiments, may be implemented as, for example, a Corba based API, an XML based API, a PARLAY server, an OSA Server, or a JAIN server.

[**0080**] Briefly, the MSSP server **22** functions as both an Internet router and an IP packet analyzer. Data contained in a header field of an IP packet is defined in the Internet Engineering Task Force (IETF) RFC 791, incorporated herein by reference (see [www.ietf.org](http://www.ietf.org)). The IETF is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet.

[**0081**] The Internet Protocol (IP) is designed for use in interconnected systems of packet-switched computer communication networks. IP provides for transmitting blocks of data called datagrams from sources to destinations, where sources and destinations are hosts identified by fixed length addresses. The IP also provides for fragmentation and reassembly of long datagrams and, if necessary, for transmission through "small packet" networks.

[**0082**] The MS SP control process **26** is designed to analyze IP packet headers in real time to manage counters and signal when packet characteristics match specified conditions. A signal may be an event report or a trigger. An event report reports an occurrence of some event while continuing to monitor packet flow. A trigger causes processing of the IP packet to be suspended until the MSSP control process **26** responds with specific instructions for resuming processing of the IP packet. A trigger response may simply direct IP packet processing to be continued unchanged, or it may alter packet processing by specifying a different destination for the packet or cause the packet to be discarded altogether. The API **28** provides, in one example, a way for the other applications **30** to communicate with the MSSP control process **26** and manipulate event reports and triggers.

[**0083**] The MSSP control process **26** manages many different types of IP packets. In one example, the MSSP control process **26** is divided into different state machines (not shown), each state machine responsible for different types of packets. In general, a state machine is any device that stores the status of something at a given time and can operate on input to change the status and/or cause an action or output to take place for any given change. In practice, state machines are used to develop and describe specific device or program interactions.

[**0084**] Within each state machine of the MSSP control process **26** there are strategic places where important information becomes available or key decisions are made. These places are called detection points. A detection point (DP) is a defined place in a state machine of a control entity where application event reporting and/or control are possible, and manageable through the API **28**.

[**0085**] An Event Detection Point (EDP) is a detection point armed within the context of an existing control dialog. Event detection points do not have explicit criteria; they are only applicable to a specific state machine instance of a

control entity that generated the control dialog. In general, event detection points set within one control dialog do not affect a behavior of any other instance of that state machine. A complete set of detection points in a given state machine is known as a detection point class.

[**0086**] One of the most commonly used Internet protocols is the Transmission Control Protocol (TCP), which is defined in IETF RFC 793. By way of example, detection points using a TCP detection point class will be discussed below. However, it should be realized that other protocols might be utilized.

[**0087**] TCP provides a reliable, connection oriented communication path between two application processes (usually referred to as a client and a server). The client initiates a connection and the server accepts the connection before any data can be exchanged. The TCP protocol ensures that all of the data sent is received by the other side correctly and in the order that it was sent.

[**0088**] In order to initiate a TCP connection to a server, a client sends an IP packet to the server's IP address containing a TCP header with a "SYN" flag set and specifying a port number of the server application that it wishes to connect to. The server accepts the connection by sending a similar SYN packet back to the client, and the client acknowledges the SYN from the server by sending an IP packet containing a TCP header with the "ACK" flag set.

[**0089**] Packets pass through the MSSP control process **26** in the MSSP server **22** on their way between a client, e.g., MS **12**, and a server (not shown) residing on the Internet **24**. By examining the IP header of the packets, the MSSP control process **26** determines that the IP packet encapsulates TCP data and assigns the packet to TCP control logic. By examining the data in the TCP header in conjunction with the data in the IP header, the TCP control logic can distinguish each segment of the connection establishment.

[**0090**] For example, suppose that one of the service applications **30** would like to "intercept" TCP connections to a specific server on the Internet **24** and redirect them to different servers on the Internet **24**, perhaps based on the service application's knowledge of current server load conditions. The service application **30** can instruct the MSSP control process **26** through the API **28** to generate a trigger that watches for a TCP SYN packet that has a destination that matches the server to be intercepted. This is referred to as an Initial Detection Point (IDP). An IDP is a detection point armed so as to generate a new control dialog with an application when conditions match given criteria.

[**0091**] All other TCP packets, and TCP SYN packets directed to a different destination, continue to be processed normally. A TCP SYN packet with a destination that matches the arming criteria, however, causes processing of that packet to be suspended and an IDP event notification sent to the service application **30** that armed the IDP through the API **28**.

[**0092**] The IDP event notification may include, for example, information from the suspended packet that the service application **30** may use to determine a correct destination for the connection. The service application **30** then directs the MSSP control process **26** through the API **28** to resume packet processing with a different destination address. The MSSP control process **26** forwards a modified

TCP SYN packet to the new destination, where that server responds in a typical manner. The service application's involvement is completely transparent, i.e., neither the client, e.g., MS 12, nor the server (not shown) on the Internet 24 is aware that any redirection has taken place.

[0093] Service applications 30 interact with the MSSP control process 26 by exchanging TCP/IP messages. The API 28 listens for connections from service applications 30. When an application connection is made, the API 28 authenticates the identity of the connected service application 30 and looks up the features that the application is authorized to access.

[0094] The service application 30, once its communication session with the API 28 is established, requests a list of services that it is expected to provide from the MSSP control process 26 and then arms Initial Detection Points needed to implement those services. After that, the service application 30 waits for the MSSP control process 26 to signal when it has a packet that matches the arming criteria.

[0095] When the MSSP control process 26 signals an IDP event, the service application 30 applies its service logic (not shown) through the API 28. This service logic may, in addition to directing the packet to a chosen destination, configure additional metering for the packet flow that encountered the detection point, request additional event reports from this flow, indicate a charge plan that is applicable to the flow, request periodic charge notification events, or request flow statistics.

[0096] In an example, using an activate service filtering request message of the API 28, a default behavior of a service interaction between the MSSP control process 26 and the service logic of the application 30 may be specified without the need to implement a trigger detection point. A source address, source port, destination address string within a data portion of a packet and protocol port are used to match incoming requests to determine if a predefined service interaction should be executed. If a flow matches the criteria, the actions specified in the remainder of the message are carried out. Example actions that may be specified include the reporting of events as well as redirecting a request to a specified redirection address and port number.

[0097] In another example, the service logic begins execution when an IDP is detected. The service logic receives an event notification that the detection point was encountered. If the detection point is registered as a request detection point, the service logic responds when the MSSP request instruction within a timeout period. The response may modify the packet then forward it, release the flow or session, or redirect or connect the packet using the connect request. Other requests may also be made to program policy filters to be applied to flow or session. Optionally the service filter request may be used by the service logic to specify the service interaction to be carried out when the detection point is encountered.

[0098] For example, the API 28 provides a connect request message that instructs the MSSP control process 26 to establish a connection to a specified destination address on a flow that is suspended at a trigger point. The destination address may be different than the destination address in the packet that matched the trigger condition. This allows the service logic in the service application 30 to, for example, route connections to a best available resource.

[0099] The API 28 provides a release flow message that instructs the MSSP control process 26 to terminate an active flow. The MSSP control process 26 will terminate the flow and may provide any events or metering messages following confirmation of the termination.

[0100] Thus, using the API 28, the service application 30 manages and controls sponsored packet switched data services, which include any and all unique network addresses that identify the packet switched data service, the policy decisions that determine how, and to which, packet switched data service provider the user is directed (e.g., a specific server on the Internet 24), and the policy decisions that determine which sponsor is to be billed for the session and on what basis. Policy filters may be used to block IP traffic in either direction based on port, protocol, IP address, cookies, or other layer seven protocol characteristics. The policy filters also allow the service logic to create and manage a wall garden or subscription based model. The policy filters are dynamic in nature, allowing new services to be purchased dynamically and updated by the service logic.

[0101] The policy decisions for selection and billing may include rules that incorporate pre-agreements between an operator and third parties, either sponsors or service providers, as to the selection of the service provider and the method and basis of payment for the sponsor. A policy decision of which service provider to make a connection to may be made at the time of the service request based upon such factors as a user identity, a location of the user, a time of day, a user class, a service provider class, network conditions, pre-agreement rules, and/or governmental regulations. For example, a policy decision of which sponsor to bill and on what basis can be made at time of the service request based upon similar factors such as the user identity, the location of the user, time of day, user class, service provider class, network conditions, pre-agreement rules, and/or governmental regulations.

[0102] A service interaction is defined by the service logic as having a beginning, middle, and end. The beginning of service interaction is typically identified by an IDP (Initial Detection Point) event sent to the service logic when the detection point is encountered. The service interaction will end when there are no further events registered by the service logic or the service logic explicitly terminates the dialogue. The service interaction is bounded by the sequence of events and API calls received and made by the service logic between the IDP and the terminal event. A service interaction is usually billable event that causes the service logic to write a CDR following the end of the interaction. The details of service interaction boundaries are determined by the service logic. A stock quote service for example may begin when an IDP matching the request is reported, and end on the response containing the quote. This same example can be expanded to include, for example, file downloads and email delivery. The MSSP provides the means to detect and control an interaction and the service logic is responsible for making the API calls and processing events to implement the service.

[0103] Referring to FIG. 2, using TCP as an example, an interception process 50 includes a service application startup stage 52, a service initialization stage 54, a service deployment stage 56, a service logic stage 58 and a shutdown stage 60.

[0104] Referring to FIG. 3, the service application startup stage 52 includes initializing (70) a transport layer. The transport layer is initialized (70) by creating a TCP/IP socket and connecting the socket through the API 28. The stage 52 initializes (72) a session layer. The initialization (72) includes sending a session open request to the MSSP server 22. The MSSP server 22 authenticates the application's credentials. A session open confirmation is received from the MSSP server 22. The stage 52 initializes (74) an application layer. The initialization (74) includes sending a negotiate API version request and receiving a negotiate API version confirmation. An open request is sent and confirmed.

[0105] Referring to FIG. 4, the service initialization stage 54 includes sending (80) a get service list request; the MSSP server 22 looks up the services for this application. The stage 54 receives (82) a get service list confirmation and sends (84) a get service detail request; the MSSP server 22 looks up configuration data for the service. The stage 54 receives (86) a get service detail request confirmation.

[0106] Referring to FIG. 5, the service deployment stage 56 includes sending (90) an Arm IDP request and receiving (92) an Arm IDP confirmation. The MSSP server 22 verifies that the arming criteria meets any restrictions configured for the application and service and programs the ICP criteria into the MSSP server 22.

[0107] Referring to FIG. 6, the service logic stage 58 includes receiving (100) an initial DP event. The stage 58 determines (102) a new destination for the subscriber connection and sends (104) a connect request to the new destination. The stage 58 receives (106) a connect confirmation.

[0108] Referring to FIG. 7, the shutdown stage 60 includes sending (110) a disarm IDP request and receiving (112) a disarm IDP confirmation. The stage 60 sends (114) a close request and receives (116) a close confirmation. The stage 60 sends (118) a session close request, receives (120) a session close confirmation, and closes (122) the TCP/IP socket.

[0109] Referring to FIG. 8, a table 130 shows a set of data types utilized to define fields within messages used by the API 28. The table 130 includes a data type name 132, a definition 134, and a byte size 136. CHAR[n] refers to a UTF-8 character string. UTF-8 is a character encoding scheme in which the entire set of ASCII characters are encoded in one byte with the same encoding as ASCII while also allowing any of the full range of Unicode characters to be encoded using multiple-byte sequences in which no byte contains an ASCII character value.

[0110] All numeric data of more than one byte in length is transmitted in a canonical network byte order defined by TCP/IP standards, i.e., in order of most significant byte to least significant byte. It should be noted that to ensure application correctness and portability, application developers are encouraged to use their platform's host-to-network and network-to-host conversion functions (such as htonl() and ntohs()) even when the host platform is known to use network byte order. htonl() is an example UNIX function that converts 32-bit (4-byte) quantities from host byte order to network byte order, while ntohs() is an example UNIX function that converts 32-bit quantities from network byte order to host byte order.

[0111] Referring to FIG. 9, a communication path 140 (indicated by the arrows) between an application program 30 and the MSSP server 22 uses a layered architecture. The application program 30 transmits data through its system's application layer 142, presentation layer 144, session layer 146, transport layer 148, TCP/IP layer 150 and lower layers 152, to corresponding lower layers 154, TCP/IP layer 156, transport layer 158, session layer 160, presentation layer 162 and application layer 164 of the MSSP SERVER 22.

[0112] The transport layer 158 is used to provide a reliable transport to the session layer 160. The transport layer 158 is relatively lightweight since it is layered on top of the local TCP/IP layer 156, which by definition is reliable. The transport layer 158 receives messages from the session layer 160 that are then transmitted. The transport layer 158 separates the byte stream provided by the TCP/IP layer 156 into messages that are framed by a transport header.

[0113] In general, a frame is data that is transmitted between network points as a unit complete with addressing and necessary protocol control information. A frame is usually transmitted serial bit by bit and contains a header field and a trailer field that "frame" the data.

[0114] Referring to FIG. 10, a representation of a TCP/IP byte stream divided into session messages by the transport layer is shown. A frame marker, unlike some other protocols, does not itself determine a boundary of a transport message header. The frame marker data pattern may also be present elsewhere in a TCP/IP byte stream with no adverse effects or special encoding. The frame marker provides a means to detect common programming errors (such as improper byte ordering or length calculation errors) that might otherwise cause a receiver to incorrectly interpret some other data as a transport message header and take inappropriate action.

[0115] The API 28 uses an 8-byte transport message header as the first element in a message. The 8-byte transport message header includes a 4-byte INIT "framemark" field that is a constant value used to verify the presence of a valid transport message header. Any other value is indicative of a message framing error. The 8-byte transport message header also includes a 4-byte "messagelength" field and contains an UNIT data type representing the length, in bytes, of the message data that follows.

[0116] The API 28 utilizes session level interfaces built on top of the reliable TCP/IP transport layer that guarantees a message will arrive. This session layer provides a set of session level services to the application layer. These services include authentication, session level heartbeats, and session level acknowledgements.

[0117] In general, a heartbeat monitors the status of a communication link and identifies when the last of a string of messages is not received. When either end of a connection has not sent any data for a specified number of seconds, it will transmit a heartbeat message. When either end of the connection has not received any data for a specified number of seconds, it will transmit a test request message. If there is still no heartbeat message received after the same time then the connection is considered lost and corrective action initiated.

[0118] All messages exchanged at the session layer include a header of four USHORT 2-byte fields as the first element in the message. The header is referred to as a session

message header and includes a SessionMessage type field, a SessionInstance field, a SessionSendSeqNo field and a SessionReceiveSeqNo field.

[0119] The SessionMessage Type field contains a value that identifies the type of message and the format of the message data. The SessionInstance field contains a value that uniquely identifies the session instance. The SessionSendSeqNo field contains the send sequence number of the message. The SessionReceiveSeqNo field contains the send sequence number from the last received message.

[0120] All session messages include a pair of sequence numbers in the Session Message Header that are set by the sender and verified by the receiver. Each sender starts at zero and increments the send sequence number for each message sent. In addition, each sender keeps track of the next SessionSendSeqNo it expects to receive. Every message sent includes this number pair. The sequence numbers are used to detect lost session messages as well as provide a means to acknowledge receipt of data. The periodic exchange of sequence numbers in session heartbeat messages ensures that the sequence numbers remain up to date in the event that the session is idle with respect to SessData messages.

[0121] The session layer protocol version is negotiated during an open sequence. A client specifies a desired version of the protocol to be used for the duration of the session. The client initially specifies the highest version of the protocol supported by the client. A server examines the requested version number and compares it against the versions it supports. If the requested version is in the range of versions supported by the server, the acceptance of the version is indicated in a subsequent SessOpenConf message. If the client has requested a version beyond those supported by the server, the server responds with a SessOpenConf message indicating that the session has been established using the highest version supported by the server. This version will be different from what was originally requested by the client. In the event that the server cannot find a mutually supported protocol version a SessError message with an error code of MSSP\_E\_INVALID\_VERSION is sent and the session is closed.

[0122] Similarly, session layer options are negotiated during the open sequence. The client specifies the desired protocol options to be used for the duration of the session. The client should always initially specify all options supported by the client. The server examines the requested options mask and chooses those options that it supports. The resulting mutual session options are communicated to the client in the subsequent SessOpenConf message. If the client is unable to function as a result of the options being reduced by the server, a SessError message with an error code of MSSP\_E\_INVALID\_OPTIONS is sent to the server and the session closed.

[0123] Similarly, a heartbeat interval is negotiated during the open sequence. The client specifies its desired heartbeat interval in the SessOpenReq message, and the server responds with the heartbeat interval that the client should use in the subsequent SessOpenConf message.

[0124] A client and server exchange credentials during a session establishment sequence. The client provides an encrypted Session Security Descriptor that is the MD5

message-digest of the SessOpenReq message (excluding the SessionSecurityDescriptor field) encrypted using a private key of a public/private key pair. The MD5 message format is designed by RSA Data Security, Inc. and defined in IETF RFC 1321 (see www.ietf.org). Since a given application will likely open its session the same way every time, a random number field is provided in the message in order to prevent generating a "constant" message digest value and a resulting predictable Session Security Descriptor. The MSSP server 22 configuration of the application contains the public key of the public/private key pair. Upon receipt of the security descriptor in a SessOpenReq message, the server looks up the application in the MSSP server 22 configuration to obtain the client's public key, decrypt the given security descriptor using the public key, and verify that the decrypted result exactly matches the MD5 message-digest generated from the received message. If the credentials fail to validate, the server responds with a SessError message with an error code of MSSP\_E\_AUTH\_FAILURE. If a number of successive failures occur in a unit amount of time, the server suspends listening for connection requests for a period of time not less than one minute.

[0125] If the credentials pass validation, the server provides a SessionSecurityDescriptor (that is the MD5 message-digest of the SessOpenConf message (excluding the SessionSecurityDescriptor field)) encrypted using the private key of a different public/private key pair) to the client in the SessOpenConf message. The client decrypts the descriptor using the server's public key and authenticates the server. If the validation of the server credentials fail on the client side of the connection, the client sends a SessError message with an error code of MSSP\_E\_AUTH\_FAILURE. If a number of successive failures occur in a unit amount of time, the client suspends connection requests for a period of time not less than one minute.

[0126] The SessOpenReq message is used to begin a session-level exchange of information between an application and the API 28. The SessOpenReq message is the first message that is after a transport layer connection has been established as described above. The SessOpenReq message has the following format:

[0127] An 8-byte SessionHeader field that is a session header with a SessionMessage Type equal to Sess\_OpenReq. A 4-byte UNIT SessionVersion field that represents a session protocol version supported by the client. A 4-byte UNIT SessionOptionsMask field that represents a bitwise combination of all the session layer options supported by the client. A 4-byte UNIT SessionHeartbeatInterval field that represents the nominal interval between exchanges of session heartbeat messages in seconds. A 4-byte UINT SessionApplicationID field that represents a MSSP server 22 determined value uniquely identifying this client application in the MSSP server 22. A 4-byte UNIT SessionRandomNum field represents any unpredictable value and is used to prevent predictable SessionSecurityDescriptor. A 16-byte BYTE[16] SessionSecurityDescriptor field representing a session security descriptor that is a MD5 message-digest of the message (excluding this field) encrypted using the client's private key of a public/private key pair. The server decrypts the session security descriptor using its copy of the client's public key to authenticate the client.

[0128] The SessOpenConf message is used to complete an establishment of a session and communicate a result of

negotiated parameters. This message is sent as the successful response to a SessOpenReq message and has the following format:

[0129] An 8-byte SessionHeader field representing a session header with SessionMessageType SESS\_OPEN\_CONF. A 4-byte UINT SessionVersion field represents a session protocol version chosen for use by the server. A 4-byte UNIT SessionOptionsMask field representing a bit-wise combination of all of the client session layer options chosen by the server. A 4-byte UNIT SessionHeartbeatInterval field representing a nominal interval between exchanges of session heartbeat messages in seconds. A 4-byte UNIT SessionServerID field represents a value uniquely identifying this MSSP SERVER 22 instance. A 4-byte UNIT SessionRandomNum field represents any unpredictable value and is used to prevent predictable SessionSecurityDescriptor. A 16-byte BYTE[16] SessionSecurityDescriptor field representing a session security descriptor that is the MD5 message-digest of the message (excluding this field) encrypted using the server's private key of a public/private key pair. The client should decrypt the session security descriptor using its copy of the server's public key to authenticate the server.

[0130] A session requires that the client and server participate in a session maintenance procedure. The session maintenance procedure ensures that inactive or idle sessions are functional as well as ensuring that the response time is within reasonable limits. The session maintenance procedure is performed independent of whether or not other data is transmitted on the session. The session maintenance procedure includes the exchange of a SessHeartbeatReq message, followed by a SessHeartbeatConf message. The session maintenance procedure is initiated from the client side of a connection by sending a SessHeartbeatReq message. The server performs a set of operations to ensure the server is functioning properly and returns a SessHeartbeatConf message if all is well. If the server fails to respond within the heartbeat interval the client fails the session by sending a SessError message with an error code of MSSP\_E\_HEARTBEAT\_TIMEOUT to the server. The client makes heartbeat requests at a periodic interval as specified in the SessOpenConf message at the time the session was established. The first client heartbeat is sent upon receiving the SessOpenConf message. Upon sending a SessHeartbeatReq message, a client timer is set to the heartbeat interval and a SessHeartbeatReq sent when the timer expires. The server expects to see a heartbeat request within the specified heartbeat interval. The server sets a timer following the transmission of the SessOpenConf message and the timeout will be set to twice the heartbeat interval. If the timer should expire and a heartbeat request is not received, the server fails the session by sending a SessError message with an error code of MSSP\_E\_HEARTBEAT\_TIMEOUT. Each time a new heartbeat request is received, the server side timer is reset. At any given instant only one heartbeat request is outstanding. Note that the heartbeat messages will also be used to acknowledge DATA messages or detect errors related to mismanagement of sequence numbers on an idle session connection.

[0131] The SessHeartbeatReq message is used to request verification that the session partner is operating normally and has the following format:

[0132] An 8-byte SessionHeader field representing a session header with SessionMessageType=SESS\_HEARTBEAT\_REQ. A 4-byte UNIT SessionHeartbeatInstance field representing a value that uniquely identifies a given heartbeat in the session. A 4-byte TIME SessionTimeStamp field represents a time that the heartbeat request was issued. A 4-byte UNIT SessionHeartbeatInterval field representing a nominal interval between exchanges of session heartbeat messages, in seconds. This may be different than the current heartbeat interval when the sender desires to negotiate a new heartbeat interval.

[0133] The SessHeartbeatConf message is used to complete the verification of the session partner's normal operational status. This message is sent as the successful response to a SessHeartbeatReq message. The SessHeartbeatConf message has the following format:

[0134] An 8-byte SessionHeader field represents a session header with SessionMessageType equal to SESS\_HEARTBEAT\_CONF. A 4-byte UNIT SessionHeartbeatInstance field represents the same SessionHeartbeatInstance value that was given in the corresponding heartbeat request. A 4-byte TIME SessionTimeStamp field represents the same SessionTimeStamp value that was given in the corresponding heartbeat request. A 4-byte UNIT SessionHeartbeatInterval field representing a nominal interval between exchanges of session heartbeat messages, in seconds. This may be different than the current heartbeat interval when a new heartbeat interval has been negotiated.

[0135] A session may be closed by either client or server at anytime following successful session establishment. A client or server initiates the closure procedure by sending a SessCloseReq message to the session partner. The SessCloseReq message contains a code indicating the reason for the closure. The requesting session partner shutdowns (in the socket sense) the transport layer following the transmission of the SessCloseReq message. The receiving session partner notifies the application layer to allow any outstanding requests on the session to be completed. Any queued session messages are sent prior to the transmission of the SessCloseConf message. Once the SessCloseConf message is sent, the transport connection shutdowns and the socket connection is closed from the side that requested the session be closed. A client may time-out a close request if a server fails to respond within a reasonable period of time. If a close request is timed-out by a client, a SessError message is sent to the server with an error code of MSSP\_E\_CLOSE\_TIMEOUT. If a session partner is unable to process a close request because a session has not been previously opened at the time of a close request, a SessError message is sent to the requesting partner with an error code of MSSP\_E\_NO\_SESSION. If a session is active or initialized and the session partner is unable to process a close request for any reason, the receiver sends a SessError message to the requester with an error code of MSSP\_E\_UNSPECIFIED\_FAILURE.

[0136] The SessCloseReq message is used to initiate the orderly termination of a session and has the following format:

[0137] An 8-byte SessionHeader field representing a session header with SessionMessageType=SESS\_CLOSE\_REQ. A 4-byte UNIT SessionCloseReasonCode field represents a value indicating a reason for the closure of the session. MSSP reason codes include, for example, normal

operation, partial detail during normal operation, normal shutdown, subscriber logout, flow timeout and session timeout.

[0138] SessCloseConf message is used to complete an orderly termination of a session. This message is sent as the successful response to a SessCloseReq message and has the following format:

[0139] An 8-byte SessionHeader field representing a session header with SessionMessageType=SESS\_CLOSE\_CONF.

[0140] One purpose of establishing a session is to exchange data between a client and a server. Data messages may be exchanged between parties following the completion of the session open sequence. The session layer does not interpret data messages. Received data messages are forwarded to the application layer for processing. Only the bytes contained in the SessionData field of a SessData message are forwarded to the application layer. This effectively removes the session portion of the message prior to passing it to the application. Messages received from the transport layer are also devoid of any transport layer headers or data, and the messages are complete prior to processing. The converse is also true when transmitting data. The session layer encapsulates the application data in a session data message that is forwarded to the transport layer for transmission.

[0141] A SessData message SessData is used to transmit application layer data to the session partner and has the following format:

[0142] An 8-byte SessionHeader field representing a session header with SessionMessageType=SESS\_DATA. A variable length SessionData field representing data to be delivered to the application layer.

[0143] A session may fail from time to time due to communication or process failures. In the event of a session failure, the failure is reported asynchronously under the context of the session partner detecting the failure. Either the client or the server side may send a SessError message. A SessError message may be sent at anytime from the client side following the SessOpenReq message. A SessError message may be sent at anytime from the server side including as a response to a SessOpenReq. A SessError message contains an error code indicating the reason for the

failure. The session partner may or may not receive the SessError message depending upon the nature of the error. Following the transmission or receipt of a SessError message, data may not be sent over the session and the underlying transport connection should be shutdown and closed.

[0144] The SessError message is used to inform a session partner of an error condition that will prevent further session level communication; it has the following format:

[0145] An 8-byte SessionHeader field representing a session header with SessionMessageType=SESS\_ERROR. A 4-byte UNIT SessionErrorCode field represents a value indicating a cause of the session failure.

[0146] Referring to FIG. 11, a table 170 containing sample error codes is shown.

[0147] Capabilities of the MSSP server 22 may be grouped into feature categories. When applications 30 open their session with the MSSP server 22 the applications 30 specify what features they want through the API 28. Each MSSP feature has a corresponding privilege bit. A configuration entry in a MSSP configuration database 32 residing in a MSSP storage device 34 for an application contains a set of feature privileges that control what features the application 30 is authorized to use. Only the requested features that are authorized for the application 30 are granted, and the application 30 is informed of the features that have successfully been obtained in the response to the request. Application attempts to use messages in a feature category that it has not been granted are refused with a privilege error.

[0148] Referring to FIG. 12, a table 180 listing feature categories is shown. Feature categories include a common services feature category 182, an Initial Detection Point feature category 184, an Event Reporting feature category 186, a Service Filter feature category 188, a Meter Configuration feature category 190, a Charge Notification feature category 192, a Charge Plane feature category 194, a Detail Record Control feature, category 196. a Statistics feature category 198, and an Application Monitor feature category 200. Messages associated with each of the feature categories 182-200, with their respective format, are listed in Appendix A, and incorporated herein by reference.

[0149] Other embodiments are within the scope of the following claims.

## **APPENDIX A**

**Common Services**

Description: The messages in this section are common to all applications using the API, regardless of application privileges.

Privileges Required: None.

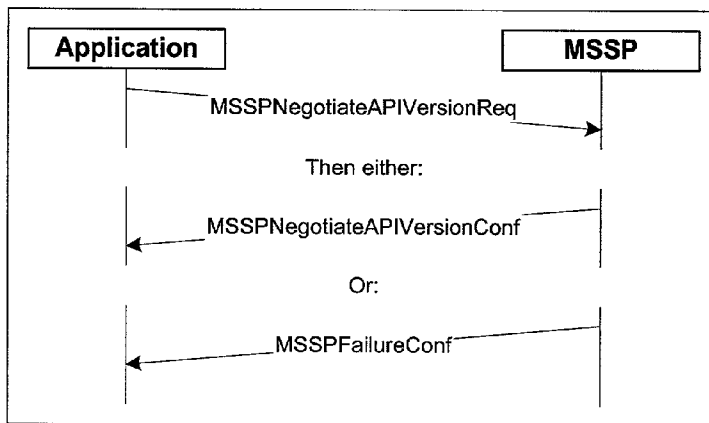
List of Messages: MSSPNegotiateAPIVersionReq, MSSPNegotiateAPIVersionConf, MSSPOpenReq, MSSPOpenConf, MSSPCloseReq, MSSPCloseConf, MSSPFailureConf, MSSPFailureEvent, MSSPGetSystemTimeReq, MSSPGetSystemTimeConf, MSSPGetServiceListReq, MSSPGetServiceListConf, MSSPGetServiceDetailReq, MSSPGetServiceDetailConf.

**MSSPNegotiateAPIVersionReq**

Description: This message is sent by the application to the MSSP 22 to indicate the API version that it would like to use for application-level communication. The API version must be negotiated before any other application messages can be exchanged since the message formats will vary. Only *MSSPNegotiateAPIVersionReq*, *MSSPNegotiateAPIVersionConf*, and *MSSPFailureConf* are guaranteed to have the same message format in all API versions. This is the first message that should be sent after a communication session has been established as described in the previous section.

The MSSP 22 replies with an *MSSPNegotiateAPIVersionConf* message specifying the negotiated API version to be used for all further application messages. This will be the highest API version supported by MSSP 22 that is less than or equal to the application requested version. Inability to identify an API version common to both parties results in an *MSSPFailureConf* message being returned from the MSSP 22 with an error code of `MSSP_E_INVALID_VERSION`.

Message Flow Diagram:



Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_NEGOTIATE_API_VERSION_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
APIVersion	The desired application protocol version to be used for subsequent messages. MSSP_API_V1 is the only protocol version defined at this time.	UINT	4

MSSPNegotiateAPIVersionReq Message Format

### MSSPNegotiateAPIVersionConf

Description: This message is sent by the MSSP 22 to acknowledge receipt of an *MSSPNegotiateAPIVersionReq* request message and provide the API version that has been chosen for all further application layer messages.

Message Format:

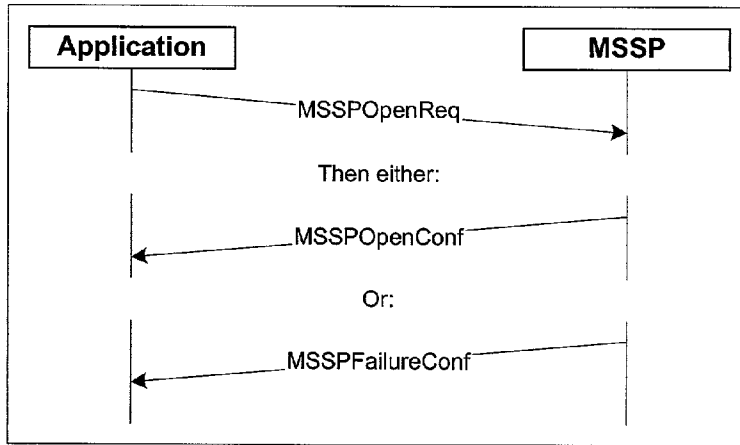
Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_NEGOTIATE_API_VERSION_CONF.	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4
APIVersion	The application protocol version chosen to be used for subsequent messages. MSSP_API_V1 is the only protocol version defined at this time.	UINT	4

MSSPNegotiateAPIVersionConf Message Format

### MSSPOpenReq

Description: This message is used to begin the application-level exchange of information between the application and the MSSP 22. This is the first message that should be sent after the API version has been established as described above. The application uses this message to request access one or more features of the MSSP 22.

Message Flow Diagram:



Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_OPEN_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
FeatureMask	A bitwise combination of one or more feature masks.	UINT	4

MSSPOpenReq Message Format

Feature Category	Feature Mask
Initial Detection Point	MSSP_FEATURE_IDP
Event Reporting	MSSP_FEATURE_EDP

Feature Category	Feature Mask
Service Filter	MSSP_FEATURE_SERVICE_FILTER
Meter Configuration	MSSP_FEATURE_METER_CONFIG
Charge Notification	MSSP_FEATURE_CHARGE_NOTIFY
Charge Plan	MSSP_FEATURE_CHARGE_PLAN
Detail Record Control	MSSP_FEATURE_DETAIL_CONTROL
Statistics	MSSP_FEATURE_STATISTICS
Application Monitor	MSSP_FEATURE_APP_MONITOR

MSSP 22 Feature Masks

### MSSPOpenConf

Description: This message is sent by the MSSP 22 to acknowledge receipt of an *MSSPOpenReq* request message. The message indicates which of the services that were requested in the *MSSPOpenReq* have actually been granted.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, <i>MSSP_OPEN_CONF</i> .	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4
FeatureMask	A bitwise combination of one or more feature masks corresponding to the features that the application is authorized to use.	UINT	4
SystemTime	The current MSSP 22 system time.	TIME	4

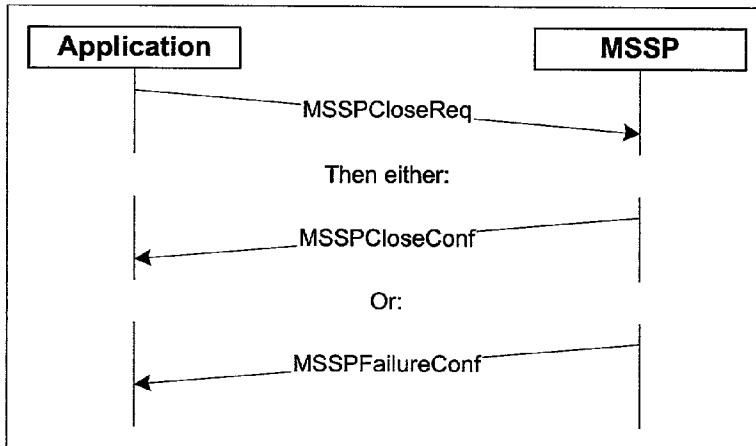
Field Name	Description	Data Type	Byte Size
MaxResources	The maximum number of MSSP 22 resources that the application is allowed to use. A value of 0xFFFF FFFF indicates no limit.	UINT	4
AppName	Application name (may be up to 32 non-null characters plus a NULL string termination character, additional message field bytes added to preserve field alignment).	CHAR[36]	36

MSSPOpenConf Message Format

### MSSPCloseReq

Description: This message is used to terminate the application-level exchange of information between the application and the MSSP 22.

Message Flow Diagram:



Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_CLOSE_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
ReasonCode	A value indicating the reason for termination.	UINT	4

MSSPCloseReq Message Format

### MSSPCloseConf

Description: This message is sent by the MSSP 22 to acknowledge receipt of an *MSSPCloseReq* request message. No further application-level messages will be sent from the MSSP 22 or accepted from the application.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_CLOSE_CONF.	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4

MSSPCloseConf Message Format

### MSSPFailureConf

Description: This message is sent by the MSSP 22 when an error condition prevents successful processing of a previous application request message. The message contains the RequestID that was specified in the application's request message as well as an error code indicating the reason for the failure.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_FAILURE_CONF.	UINT	4
RequestID	The RequestID value from the failing request message.	UINT	4
SubjectMessageType	The MessageType value of the failing request message.	UINT	4
ErrorCode	A value indicating the nature of the failure.	UINT	4

MSSPFailureConf Message Format

### MSSPFailureEvent

Description: This message is sent by the MSSP 22 when an error condition occurs that is not directly associated with the processing of a previous application request message. The message contains an error code indicating the reason for the failure.

Message Format:

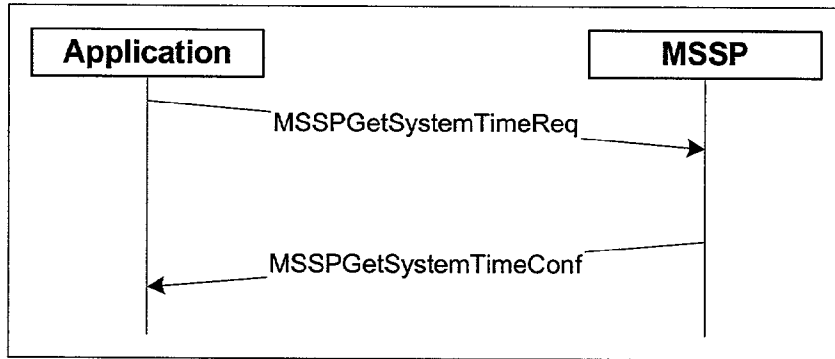
Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_FAILURE_EVENT.	UINT	4
ErrorCode	A value indicating the nature of the failure.	UINT	4

MSSPFailureEvent Message Format

### MSSPGetSystemTimeReq

Description: This message is used to request the current time from the MSSP 22.

Message Flow Diagram:



Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_GET_SYSTEM_TIME_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4

MSSPGetSystemTimeReq Message Format

### MSSPGetSystemTimeConf

Description: This message is sent by the MSSP 22 in response to an *MSSPGetSystemTimeReq* request message.

Message Format:

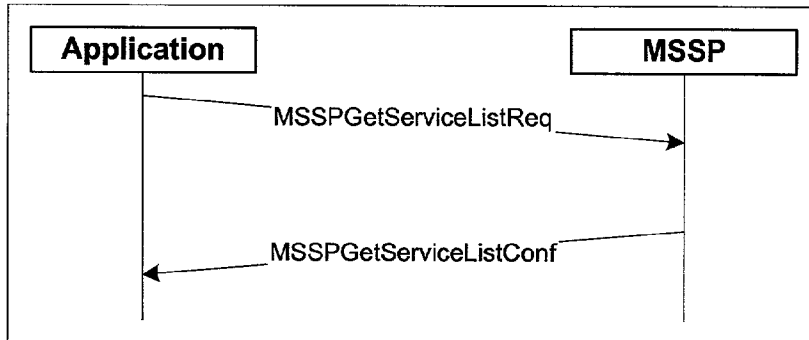
Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_GET_SYSTEM_TIME_CONF.	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4
SystemTime	The current MSSP 22 system time.	TIME	4

MSSPGetSystemTimeConf Message Format

### MSSPGetServiceListReq

Description: This message is used to request the list of MSSP 22 service identifiers that the application has been configured to provide.

Message Flow Diagram:



Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_GET_SERVICE_LIST_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4

MSSPGetServiceListReq Message Format

### MSSPGetServiceListConf

Description: This message is sent by the MSSP 22 in response to an *MSSPGetServiceListReq* request message.

Message Format:

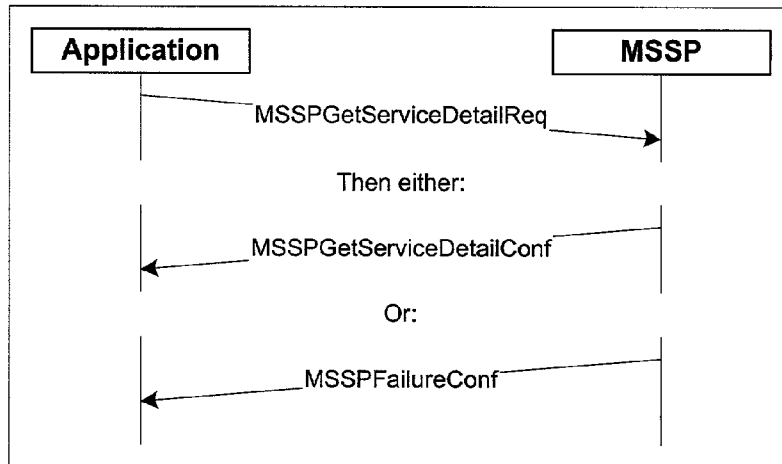
Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_GET_SERVICE_LIST_CONF.	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4
	(alignment filler)	USHORT	2
NumServices	The number of ServiceIDs that are present in the following ServiceID list.	USHORT	2
ServiceIDList	An array of MSSP 22 ServiceID values of the services that this application is configured to provide.	UINT [*]	4 * NumServices

MSSPGetServiceListConf Message Format

### MSSPGetServiceDetailReq

Description: This message is used to request detailed information about the configuration of a specified MSSP 22 service. The application may only request details for services that it is configured to provide.

Message Flow Diagram:



Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_GET_SERVICE_DETAIL_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
ServiceID	The MSSP 22 ServiceID value of the service whose details are requested. The service must be one of the services that this application is configured to provide.	UINT	4

MSSPGetServiceDetailReq Message Format

### MSSPGetServiceDetailConf

Description: This message is sent by the MSSP 22 in response to an *MSSPGetServiceDetailReq* request message.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_GET_SERVICE_DETAIL_CONF.	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4
ServiceID	The MSSP 22 ServiceID value of the service.	UINT	4
ServiceName	The name of the MSSP 22 service (may be up to 32 non-null characters plus a NULL string termination character, additional message field bytes added to preserve field alignment).	CHAR[36]	36
MaxSessions	The maximum total number of simultaneous subscriber sessions allowed to be using this service.	UINT	4
MaxFlows	The total number of simultaneous flows allowed to be using this service.	UINT	4
BillingPlanID	A value used by external billing systems for charging purposes.	UINT	4
EnabledMeterMask	A bitwise combination of meter masks representing metering elements that are automatically configured when a flow is associated with this service.	UINT	4
DetectionPointClass	A non-zero value indicates that the service is restricted to detection points in the given class only.	UINT	4
OperatorID	A non-zero value indicates that OperatorID detection point criteria is restricted to the given value for this service.	UINT	4
SubscriberGroupID	A non-zero value indicates that SubscriberGroupID detection point criteria is restricted to the given value for this service.	UINT	4

Field Name	Description	Data Type	Byte Size
SourceIPType	One of the symbolic constants: IP_NONE, IP_V4, IP_V6.	USHORT	2
SourceIPLength	Length of SourceIPAddress (in bits) that must be matched. For example, the IP_V4 address 151.104.0.0 with a length of 16 would restrict the SourceIPAddress criteria to IP addresses beginning with 151.104.	USHORT	2
SourceIPAddress	If SourceIPType is not IP_NONE then SourceIPAddress detection point criteria is restricted to the given value for this service.	BYTE[16]	16
SourcePort	A non-zero value indicates that SourcePort detection point criteria is restricted to the given value for this service.	UINT	4
DestinationIPType	One of the symbolic constants: IP_NONE, IP_V4, IP_V6.	USHORT	2
DestinationIPLength	Length of DestinationIPAddress (in bits) that must be matched (as described in SourceIPLength, above).	USHORT	2
DestinationIPAddress	If DestinationIPType is not IP_NONE then DestinationIPAddress detection point criteria is restricted to the given value for this service.	BYTE[16]	16
DestinationPort	A non-zero value indicates that DestinationPort detection point criteria is restricted to the given value for this service.	UINT	4

MSSPGetServiceDetailConf Message Format

**MSSPServiceRemovedEvent**

Description: This message is sent by the MSSP 22 when a service is removed from the list of services that the application is configured to provide while the application is connected to the MSSP 22. Any service resources (such as detection points) used by the application are automatically released by the MSSP 22.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_SERVICE_REMOVED_EVENT.	UINT	4
ServiceID	The MSSP 22 ServiceID value of the service.	UINT	4

MSSPServiceRemovedEvent Message Format

### MSSPResourceUnavailableEvent

Description: This message is sent by the MSSP 22 when failure conditions or MSSP 22 hardware reconfiguration has caused a resource used by the application to become unavailable. An *MSSPResourceAvailableEvent* message will be sent when the resource is restored to normal operation.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_RESOURCE_UNAVAILABLE_EVENT.	UINT	4
ResourceID	The unique identifier of the resource that has become unavailable.	UINT	4

MSSPResourceUnavailableEvent Message Format

### MSSPResourceAvailableEvent

Description: This message is sent by the MSSP 22 when a resource that was previously reported unavailable in an *MSSPResourceUnavailableEvent* has been restored to normal operation.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_RESOURCE_AVAILABLE_EVENT.	UINT	4
ResourceID	The unique identifier of the resource that has been restored to normal operation.	UINT	4

MSSPResourceAvailableEvent Message Format

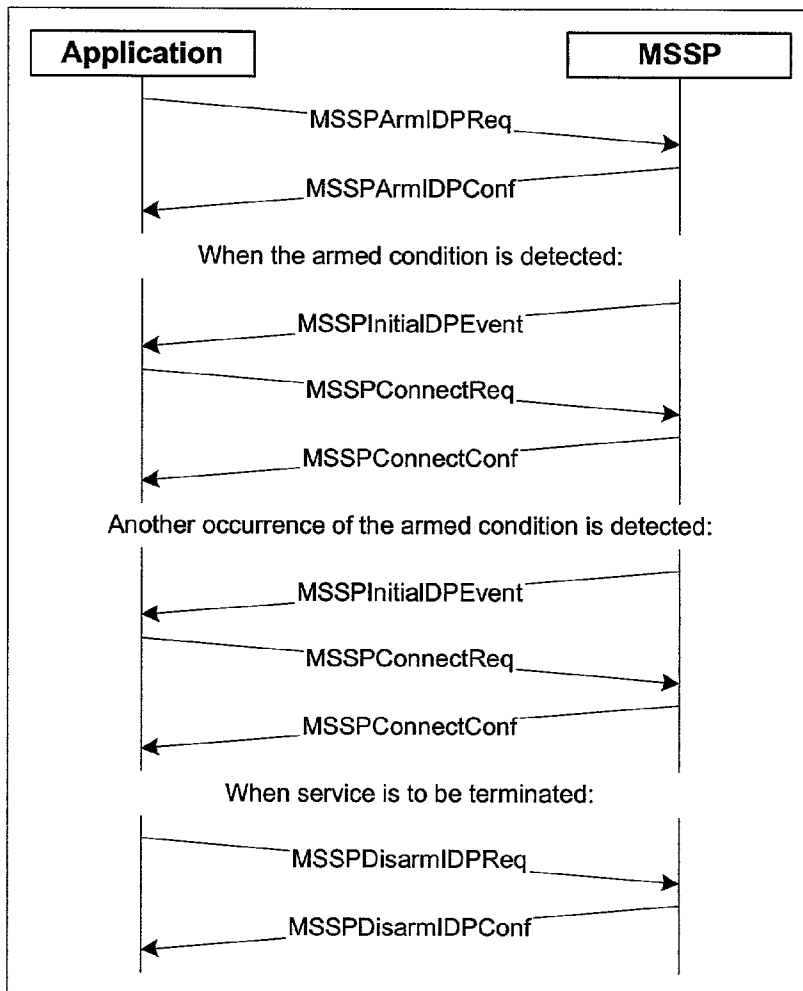
### Initial Detection Point (IDP) Feature

Description: The messages in this section allow an application to arm and disarm Initial Detection Points in the MSSP 22 and service the associated IDP events.

Privileges Required: IDP.

List of Messages: MSSPArmIDPReq, MSSPArmIDPConf, MSSPDisarmIDPReq, SSPDisarmIDPConf, MSSPInitialDPEvent, MSSPContinueReq, MSSPContinueConf, MSSPConnectReq, MSSPConnectConf, MSSPReleaseReq, MSSPReleaseConf, MSSPActivityTestReq, MSSPActivityTestConf.

Message Flow Diagram:



**MSSPArmIDPReq**

Description: This request is used to identify an initial detection point and specify the traffic criteria that should cause an application to be notified. The initial detection point may be armed for simple event notification or as a trigger, depending upon the setting of the TakeControl field. Setting the TakeControl field to MSSP\_TRIGGER arms a trigger.

When a flow encounters a detection point with a trigger, packet forwarding is halted and the application is notified. The application controls the resumption of packet forwarding by responding with one of the following requests:

*MSSPContinueReq,*  
*MSSPConnectReq,*  
*MSSPControlReq,* or  
*MSSPReleaseReq.*

These requests instruct the MSSP 22 how to proceed and packets in the flow will be delivered accordingly. Applications must respond promptly to triggers. The delay time between a trigger and the associated response is measured by service and by application, and failure to respond to a trigger within 1000 ms. will cause the MSSP 22 to increment the service and application trigger timeout counters and resume normal packet processing as if a "continue" response was received. If the detection point was armed only for event notification, event notification is sent to the application as in the trigger case, except that packet forwarding is not halted and no response is expected from the application.

The criteria strings may contain wildcard values that may be used to specify a wide range of triggers. The MSSP 22 sends an *MSSPArmIDPConf* message after the IDP is successfully armed. In the event that error conditions prevent arming the IDP an *MSSPFailureConf* message is returned instead indicating the reason for the failure.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_ARM_IDP_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
ServiceID	The identifier of the MSSP 22 service that this detection point is to be associated with.	UINT	4
DetectionPointClass	One of the values from the table below, identifying the detection point class that	UINT	4

Field Name	Description	Data Type	Byte Size
	contains the detection point being armed.		
DetectionPoint	One of the values from the appropriate detection point class table specifying the specific detection point to be armed.	UINT	4
TakeControl	A value determining whether the detection point is to be armed for event reporting (MSSP_ARM_EVENT_REPORT) or as a trigger (MSSP_ARM_TRIGGER).	UINT	4
PacketDataCopyBytes	The number of bytes to be copied from the packet matching the criteria into the IDP Event Indication message. The value 0xFFFFFFFF may be used to request a copy of the entire IP packet (not including the media-specific layer2 header).	UINT	4
OperatorID	Operator criteria. The value 0xFFFFFFFF is a wild card that matches any OperatorID.	UINT	4
SubscriberGroupID	SubscriberGroup criteria. The value 0xFFFFFFFF is a wild card that matches any SubscriberGroupID.	UINT	4
SessionID	SessionID criteria. The value 0xFFFFFFFF is a wild card that matches any SessionID.	UINT	4
SourceIPType	One of the symbolic constants: IP_NONE, IP_V4, IP_V6	USHORT	2
SourceIPLength	Length of SourceIPAddress (in bits) to be matched. For example, providing the IP_V4 address 151.104.0.0 with a length of 16 would match all IP addresses beginning with 151.104. A length of zero specifies a wildcard that matches any address.	USHORT	2
SourceIPAddress	Source IPAddress criteria.	BYTE[16]	16

Field Name	Description	Data Type	Byte Size
SourcePort	Source port criteria. The value 0xFFFFFFFF is a wild card that matches any port number.	UINT	4
DestinationIPType	One of the symbolic constants: IP_NONE, IP_V4, IP_V6	USHORT	2
DestinationIPLength	Length of DestinationIPAddress (in bits). Wild cards may be specified as described for SourceIPLength, above.	USHORT	2
DestinationIPAddress	Destination IPAddress criteria.	BYTE[16]	16
DestinationPort	Destination port criteria. The value 0xFFFFFFFF is a wild card that matches any port number.	UINT	4
NumericValue1	A numeric value criteria (IP protocol number, counter value, etc.)	UINT	4
NumericValue2	A numeric value criteria (IP protocol number, counter value, etc.)	UINT	4
StringValue	A string value criteria (hostname, username, etc.).	VARCHAR	*

MSSPArmiDPRReq Message Format

Detection Point Class	Description
DP_CLASS_SESS_GROUP	Session Group state machine.
DP_CLASS_SESSION	Subscriber session state machine.
DP_CLASS_RADIUS	RADIUS protocol state machine.
DP_CLASS_DHCP	DHCP protocol state machine.
DP_CLASS_DNS	DNS protocol state machine.
DP_CLASS_TCP	TCP protocol state machine.
DP_CLASS_IP	IP protocol state machine.

Detection Point Classes

The following sections describe each of these detection point classes in further detail. In each section the list of detection points available in the class is enumerated along with the attributes and arming criteria associated with each detection point. Detection points with "IDP" attributes may be armed as an initial detection point. Detection points with "Trigger" attributes may be armed as either a trigger or an event report. Detection points that are not listed with the "Trigger" attribute are only capable of providing event reports

### Session Group Detection Point Class

This class of detection points allows an application to implement policy decisions when various subscriber group limits are exceeded. The application provides the limiting values in the IDP arming criteria. If the IDP is armed as a trigger, the application may decide whether to allow the limit to be exceeded or not by sending Continue or Release trigger responses, respectively.

Detection Point	Description	Attributes	Criteria
DP_SESS_GROUP _SLA_EXCEEDED	Service Level Agreement has been exceeded.	IDP	OperatorID, SubscriberGroupID, NumericValue1=BurstLength, NumericValue2=MaxBandwidth
DP_SESS_GROUP _SESS_LIMIT_EXCEEDED	The maximum number of concurrent subscriber sessions has been exceeded.	IDP, Trigger	OperatorID, SubscriberGroupID, NumericValue1=MaxNumSessions
DP_SESS_GROUP _FLOW_LIMIT_EXCEEDED	The maximum number of concurrent subscriber session flows has been exceeded.	IDP, Trigger	OperatorID, SubscriberGroupID, NumericValue1=MaxNumFlows

Session Group Class Detection Points

**IDP Event Notes:** The NumericValue1 and NumericValue2 parameters in the Initial DP Event message contain the actual values of the parameters when the detection point was evaluated, not the limit values.

**Criteria Notes:** All criteria must be fully specified, no wild-card values are accepted.

**Control Operations:** None.

### Session Detection Point Class

This class of detection points allows an application to monitor and control the establishment and termination of mobile subscriber sessions. If the IDP is armed as a trigger, the application may decide whether to allow the subscriber session to proceed or not by sending Continue or Release trigger responses, respectively.

<b>Detection Point</b>	<b>Description</b>	<b>Attributes</b>	<b>Criteria</b>
DP_SESS_CREATE_COMPLETE	A new mobile subscriber session is being created.	IDP, Trigger	OperatorID, SubscriberGroupID, StringValue=Subscriber
DP_SESS_TIME_EXPIRATION	The maximum time limit for a mobile subscriber session has been reached.	IDP	OperatorID, SubscriberGroupID, StringValue=Subscriber
DP_SESS_RELEASE_COMPLETE	A mobile subscriber session has been terminated.	IDP	OperatorID, SubscriberGroupID, StringValue=Subscriber

Session Class Detection Points

Criteria Notes: Wild-card values may be specified for OperatorID and SubscriberGroupID. A zero-length StringValue may be specified as a wild-card that matches any subscriber.

Control Operations: None.

### **RADIUS Detection Point Class**

This class of detection points allows an application to monitor and control the Remote Authentication Dial In User Service (RADIUS) protocol activity of mobile subscriber sessions. If the IDP is armed as a trigger, the application may issue control operations to control subscriber acceptance and alter RADIUS message attributes.

Detection Point	Description	Attributes	Criteria
DP_RADIUS_AUTH_REQ	An Auth Request is being sent for the subscriber.	IDP, Trigger	OperatorID, SubscriberGroupID, StringValue=Subscriber
DP_RADIUS_ACCESS_ACCEPT	An Access Accept is being sent for the subscriber.	IDP, Trigger	OperatorID, SubscriberGroupID, StringValue=Subscriber
DP_RADIUS_ACCESS_REJECT	An Access Reject has been sent for the subscriber.	IDP	OperatorID, SubscriberGroupID, StringValue=Subscriber
DP_RADIUS_ACCOUNTING_STOP	An Accounting Request with STOP set has been sent for the subscriber, indicating subscriber logout.	IDP, Trigger	OperatorID, SubscriberGroupID, StringValue=Subscriber

RADIUS Class Detection Points

Criteria Notes: Wild-card values may be specified for OperatorID and SubscriberGroupID. A zero-length StringValue may be specified as a wild-card that matches any subscriber.

Control Operations: The following control operations are defined:

OperationCode	Description	Operands
OP_RADIUS_RETURN_ACCEPT	Return Access Accept to subscriber with provided attributes.	RADIUS attribute id and value pairs to be included in the Accept packet.
OP_RADIUS_RETURN_REJECT	Return Access Reject to the subscriber.	None.
OP_RADIUS_STRIP_ATTRIBUTE	Remove attributes matching the provided attribute id(s) from the packet.	RADIUS attribute ids.
OP_RADIUS_ADD_ATTRIBUTE	Add the provided attribute(s) to the packet.	RADIUS attribute id and value pairs to be included in the packet.

**DHCP Detection Point Class**

This class of detection points allows an application to monitor and control the Dynamic Host Configuration Protocol (DHCP) activity of mobile subscriber sessions.

<b>Detection Point</b>	<b>Description</b>	<b>Attributes</b>	<b>Criteria</b>
DP_DHCP_DISCOVER	A Discover Request is being sent from the subscriber.	IDP, Trigger	OperatorID, StringValue=Subscriber
DP_DHCP_ACK	An Ack indicating completion of IP address assignment has been sent to the subscriber.	IDP	OperatorID, StringValue=Subscriber
DP_DHCP_LEASE_RENEW	A Lease Renewal is being sent from the subscriber.	IDP, Trigger	OperatorID, StringValue=Subscriber
DP_DHCP_RELEASE_REQ	A Release Request has been sent from the subscriber.	IDP	OperatorID, StringValue=Subscriber

DHCP Class Detection Points

**IDP Event Notes:** The DestinationIP parameters in the Initial DP Event message from the DP\_DHCP\_ACK detection point contain the IP address being assigned to the subscriber.

**Criteria Notes:** Wild-card values may be specified for OperatorID. A zero-length StringValue may be specified as a wild-card that matches any subscriber.

**Control Operations:** None.

**DNS Detection Point Class**

This class of detection points allows an application to monitor and control the Domain Name System (DNS) protocol activity of mobile subscriber sessions. Control operations are defined that allow an application to provide an IP address to resolve a mobile subscriber DNS query.

Detection Point	Description	Attributes	Criteria
DP_DNS_QUERY_RECEIVED	DNS query is being sent by subscriber.	IDP, Trigger	OperatorID, SubscriberGroupID, SessionID, StringValue=hostname
DP_DNS_QUERY_RESPONSE	DNS query response is being sent to subscriber.	IDP, Trigger	OperatorID, SubscriberGroupID, SessionID, StringValue=hostname

DNS Class Detection Points

IDP Event Notes: The DestinationIP parameters in the Initial DP Event message from the DP\_DNS\_QUERY\_RESPONSE detection point contain the IP address returned from the DNS server.

Criteria Notes: Wild-card values may be specified for OperatorID and SubscriberGroupID. StringValue may specify a partial wild-card of the form *“.domain”*, such as *“.yahoo.com”*. A zero-length StringValue may also be specified as a wild-card that matches any hostname.

Control Operations: The following control operations are defined:

OperationCode	Description	Operands
OP_DNS_USE_ADDRESS	Return the given IP address as the DNS Query Response to the subscriber.	IP address.
OP_DNS_USE_HOSTNAME	Substitute the given hostname in place of the subscriber-specified hostname in the DNS Query.	Hostname.

DNS Control Operations

**TCP Detection Point Class**

This class of detection points allows an application to monitor and control the Transmission Control Protocol (TCP) activities of mobile subscriber sessions.

<b>Detection Point</b>	<b>Description</b>	<b>Attributes</b>	<b>Criteria</b>
DP_TCP_FORWARD_SYN	A TCP SYN packet is being sent from the subscriber to a server, initiating a connection.	IDP, Trigger	OperatorID, SubscriberGroupID, SessionID, SourceIPAddress, DestinationIPAddress, SourcePort, DestinationPort
DP_TCP_REVERSE_SYN	A TCP SYN packet is being sent from a server to the subscriber, indicating connection acceptance.	IDP, Trigger	OperatorID, SubscriberGroupID, SessionID, SourceIPAddress, DestinationIPAddress, SourcePort, DestinationPort
DP_TCP_ACK	A TCP ACK packet is being sent from the subscriber to a server, completing establishment of the connection.	IDP, Trigger	OperatorID, SubscriberGroupID, SessionID, SourceIPAddress, DestinationIPAddress, SourcePort, DestinationPort
DP_TCP_FORWARD_FIN	A TCP FIN packet has been sent from the subscriber to a server, indicating closure of the client side of the connection.	IDP	OperatorID, SubscriberGroupID, SessionID, SourceIPAddress, DestinationIPAddress, SourcePort, DestinationPort
DP_TCP_REVERSE_FIN	A TCP FIN packet has been sent from a server to the subscriber, indicating closure of the server side of the connection.	IDP	OperatorID, SubscriberGroupID, SessionID, SourceIPAddress, DestinationIPAddress, SourcePort, DestinationPort
DP_TCP_RESET	A TCP RESET packet has been sent, indicating connection reset.	IDP	OperatorID, SubscriberGroupID, SessionID, SourceIPAddress, DestinationIPAddress, SourcePort, DestinationPort

TCP Class Detection Points

Criteria Notes: Wild-card values may be specified for OperatorID, SubscriberGroupID, SessionID, SourcePort, and DestinationPort. SourceIPAddress and DestinationIPAddress may be specified as a partial wild-card IP address by specifying the number of address bits that must match (from left-to-right). A zero-length IPAddress may be specified as a wild-card that matches any IP address.

Control Operations: None.

**IP Detection Point Class**

This class of detection points allows an application to monitor and control the activities of mobile subscriber sessions at the lowest level, the Internet Protocol (IP) layer that all other protocols are layered on top of.

Detection Point	Description	Attributes	Criteria
DP_IP_PACKET	An IP data packet is being transmitted to or from the mobile subscriber.	IDP, Trigger	OperatorID, SubscriberGroupID, SessionID, SourceIPAddress, DestinationIPAddress, NumericValue1=IPProtocolNumber.

IP Class Detection Points

Criteria Notes: Wild-card values may be specified for OperatorID, SubscriberGroupID, SessionID, and IPProtocolNumber (NumericValue1). SourceIPAddress and DestinationIPAddress may be specified as a partial wild-card IP address by specifying the number of address bits that must match (from left-to-right). A zero-length IPAddress may be specified as a wild-card that matches any IP address. A zero-length StringValue may be specified as a wild-card that matches any subscriber.

Extreme care must be exercised when arming detection points at this level, especially the use of wild-card values, to avoid severely impacting network performance (the application becomes the bottleneck that limits the throughput of all internet traffic). While it is possible to arm a trigger with all criteria specified as wild-cards, doing so would clearly be inappropriate in an operational environment and is highly discouraged.

Control Operations: None.

### MSSPArmIDPConf

Description: This message is sent by the MSSP 22 to acknowledge successful arming of an Initial Detection Point by a previous *MSSPArmIDPReq* message.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, <i>MSSP_ARM_IDP_CONF</i> .	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4
ResourceID	An MSSP 22 determined value that uniquely identifies the newly established detection point and arming criteria set. This value is required in order to disarm this IDP.	UINT	4

MSSPArmIDPConf Message Format

### MSSPDisarmIDPReq

Description: This request is used to disarm an initial detection point, causing a previously established set of traffic criteria to be discarded.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, <i>MSSP_DISARM_IDP_REQ</i> .	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
ResourceID	The MSSP 22 determined value that uniquely identifies the detection point and arming criteria set to be discarded, as returned in the <i>MSSPArmIDPConf</i> message that armed the IDP.	UINT	4

MSSPDisarmIDPReq Message Format

**MSSPDisarmIDPConf**

Description: This message is sent by the MSSP 22 to acknowledge successful disarming of an Initial Detection Point by a previous *MSSPDisarmIDPReq* message.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, <i>MSSP_DISARM_IDP_CONF</i> .	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4

MSSPDisarmIDPConf Message Format

**MSSPInitialDPEvent**

Description: The initial detection point event is used to indicate that the conditions described by the criteria at a previously armed initial detection point have been met. Detection points are armed in order to get visibility or control of data flows matching a particular pattern. The IDP Event Indication provides fully qualified data for all of the criteria relevant to that detection point whether or not wild cards were used in the arming criteria.

Initial Detection Points that have been armed with the TakeControl option set are known as triggers. The initial detection point event sent to the associated application to indicate that a trigger or an event detection point has been hit will indicate if the detection point is due to a trigger detection point by setting the TakeControl flag to *MSSP\_TRIGGER* in the event message.

The application must respond to a trigger detection event by sending one of the following requests:

*MSSPContinueReq*, *MSSPConnectReq*, *MSSPControlReq*, or *MSSPReleaseReq*.

No response is required or expected for non-trigger Initial Detection Point Event messages.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, <i>MSSP_INITIAL_DP_EVENT</i> .	UINT	4
DETECTIONPOINTCLASS	One of the values from on page 19 identifying the detection point class that contains the detection point	UINT	4

Field Name	Description	Data Type	Byte Size
	that caused this notification.		
DetectionPoint	One of the values from the appropriate detection point class table (pages 20 - 26) identifying the specific detection point that caused this notification.	UINT	4
ResourceID	The MSSP-determined value that uniquely identifies the detection point criteria set that was matched. This is the same value that was returned in the <i>MSSPArmIDPConf</i> message.	UINT	4
ControlID	An MSSP-determined value that uniquely identifies the control dialog created by this IDP event.	UINT	4
TakeControl	A value indicating whether the flow is suspended at a trigger awaiting further control (MSSP_TRIGGER) or not (MSSP_EVENT_REPORT).	UINT	4
OPERATORID	Operator (when applicable to this detection point).	UINT	4
SUBSCRIBERGROUPID	SubscriberGroup (when applicable to this detection point).	UINT	4
SessionID	SessionID (when applicable to this detection point).	UINT	4
FlowID	An MSSP-determined value that uniquely identifies the logical data path through the control logic that encountered the detection point.	UINT	4
SourceIPType	One of the symbolic constants: IP_NONE, IP_V4, IP_V6	USHORT	2
SourceIPLength	Length of SourceIPAddress (bits).	USHORT	2
SourceIPAddress	Source IPAddress (when applicable to this detection point).	BYTE[16]	16
SourcePort	Source port (when applicable	UINT	4

A-29

Field Name	Description	Data Type	Byte Size
	to this detection point).		
DestinationIPType	One of the symbolic constants: IP_NONE, IP_V4, IP_V6	USHORT	2
DestinationIPLength	Length of DestinationIPAddress (bits).	USHORT	2
DestinationIPAddress	Destination IPAddress (when applicable to this detection point).	BYTE[16]	16
DestinationPort	Destination port (when applicable to this detection point).	UINT	4
NumericValue1	A numeric value (IP protocol number, counter value, etc.), when applicable to this detection point.	UINT	4
NumericValue2	A numeric value (IP protocol number, counter value, etc.), when applicable to this detection point.	UINT	4
STRINGVALUE	A string value criteria (hostname, username, etc.), when applicable to this detection point.	VARCHAR	*
PacketData	Binary data extracted from the packet (if requested).	VARBYTE	*

MSSPInitialDPEvent Message Format

### MSSPContinueReq

**Description:** The continue request will cause normal processing to resume on a packet that was previously suspended at a trigger point. This request might be used to provide an application synchronization point where the application can pace connection requests. The packet to be continued and its associated context is identified by the ControlID field within the request message.

If the ControlID is not valid an *MSSPFailureConf* message will be sent as a confirmation with a failure code of *MSSP\_E\_INVALID\_CONTROL\_ID*. If the ControlID is valid but not waiting at a trigger detection point, an *MSSPFailureConf* message will be sent as a confirmation with a failure code of *MSSP\_E\_INVALID\_STATE*. If the continuation operation is successful an *MSSPContinueConf* will be sent to positively confirm that packet processing has been continued.

## Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_CONTINUE_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
ControlID	The ControlID of the control dialog that was created by the corresponding IDP event.	UINT	4
ControlFlags	A bitwise combination of Control Flag values from below.	UINT	4
PACKETDATACOPYBYTES	The number of bytes to be copied from the packet at the requested event report detection point. The value 0xFFFFFFFF may be used to request a copy of the entire IP packet (not including the media-specific layer2 header).	UINT	4
EventReportMask	A bitwise combination of values from the appropriate detection point class table identifying the specific detection point(s) that should be armed for event reporting during this control dialog. Requires EDP privileges in addition to IDP privileges.	UINT	4

Field Name	Description	Data Type	Byte Size
TriggerMask	A bitwise combination of values from the appropriate detection point class table identifying the specific detection point(s) that should be armed as a trigger during this control dialog. Requires EDP privileges in addition to IDP privileges.	UINT	4

MSSPContinueReq Message Format

Control Flag	Description
CONTROL_DISABLE_VIP	Disable virtual IP address allocation.
CONTROL_REMAP_FLOW	“Hint” indicating that there is no further interest in the activity of this flow. May be used to allow reprogramming of MSSP 22 hardware for best performance.

Control Flags

**MSSPContinueConf**

Description: This message is sent by the MSSP 22 to acknowledge successful continuation of packet processing by a previous *MSSPContinueReq* message.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, <i>MSSP_CONTINUE_CONF</i> .	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4

MSSPContinueConf Message Format

### MSSPConnectReq

Description: The connect request will instruct the MSSP 22 to establish a connection to the specified destination address on a packet that was previously suspended at a trigger point. The destination address may be different than the destination address in the packet that matched the trigger condition. This will allow the application to route connections to the best available resource as well as supply a means for virtualization of Packet800 services.

The suspended packet and its associated context is identified by the ControlID field within the request message, and the destination fields will provide IP address and port number on which the connection will be established. The EventReportMask and TriggerMask can be used to request subsequent event reports and triggers from this instance of the detection point class.

If the ControlID is not valid an *MSSPFailureConf* message will be sent as a confirmation with a failure code of *MSSP\_E\_INVALID\_CONTROL\_ID*. If the ControlID is valid but not waiting at a trigger detection point, an *MSSPFailureConf* message will be sent as a confirmation with a failure code of *MSSP\_E\_INVALID\_STATE*. If the connect operation is successful, an *MSSPConnectConf* will be sent to positively confirm that packet processing has resumed.

#### Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_CONNECT_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
ControlID	The ControlID of the control dialog that was created by the corresponding IDP event.	UINT	4
ControlFlags	A bitwise combination of Control Flag values.	UINT	4

<b>Field Name</b>	<b>Description</b>	<b>Data Type</b>	<b>Byte Size</b>
DestinationIPType	One of the symbolic constants: IP_NONE, IP_V4, IP_V6	USHORT	2
DestinationIPLength	Length of DestinationIPAddress (bits).	USHORT	2
DestinationIPAddress	Destination IPAddress.	BYTE[16]	16
DestinationPort	Destination port number.	UINT	4
VPNID	The VPN or TunnelID that should be used for the connection.	UINT	4
PACKETDATACOPYBYTES	The number of bytes to be copied from the packet at the requested event report detection point. The value 0xFFFFFFFF may be used to request a copy of the entire IP packet (not including the media-specific layer2 header).	UINT	4
EventReportMask	A bitwise combination of values from the appropriate detection point class table (pages 20 - 26) identifying the specific detection point(s) that should be armed for event reporting during this control dialog. Requires EDP privileges in addition to IDP privileges.	UINT	4

10010 020001

Field Name	Description	Data Type	Byte Size
TriggerMask	A bitwise combination of values from the appropriate detection point class table (pages 20 - 26) identifying the specific detection point(s) that should be armed as a trigger during this control dialog. Requires EDP privileges in addition to IDP privileges.	UINT	4

MSSPConnectReq Message Format

### MSSPConnectConf

Description: This message is sent by the MSSP 22 to acknowledge successful execution of a previous *MSSPConnectReq* message.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_CONNECT_CONF.	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4

MSSPConnectConf Message Format

### MSSPControlReq

Description: This message will be issued to perform control operations upon the suspended packet. The suspended packet and its associated context is identified by the ControlID field within the request message. If the ControlID is not valid an *MSSPFailureConf* message will be sent as a confirmation with a failure code of MSSP\_E\_INVALID\_CONTROL\_ID. If the ControlID is valid but not waiting at a trigger detection point, an *MSSPFailureConf* message will be sent as a confirmation with a failure code of MSSP\_E\_INVALID\_STATE. If the ControlID is valid and waiting at a trigger detection point but the detection point class does not support the requested control operation, an *MSSPFailureConf* message will be sent as a confirmation with a failure code of MSSP\_E\_INVALID\_CONTROL\_OP. An *MSSPControlConf* will be sent if the control operation is successful.

This section specifies the generic definition of this message that is shared by all detection point classes. The remainder of the message content is specific to each detection point class. These detection point class-specific fields follow a common message formatting scheme: each field is identified by a two-byte tag, followed by a two-byte length field that specifies the byte size of the following data, followed by the

data. Each additional message field is simply appended to the message. The overall length of the message (provided by the underlying transport mechanism) can be used to determine the presence of these "floating" fields.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_CONTROL_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
ControlID	The ControlID of the control dialog that was created by the corresponding IDP event.	UINT	4
ControlFlags	A bitwise combination of Control Flag values.	UINT	4
OperationCode	The type of control operation to be performed. Control operations are specific to each detection point class. The format and content of the rest of the message is variable.	UINT	4
<i>[ControlTag1]</i>	One of the values from the table below, identifying the associated control data field.	USHORT	2
<i>[ControlLength1]</i>	Length of following data, in bytes.	USHORT	2
<i>[ControlData1]</i>	The data field value.	*	*
<i>[ControlTag2]</i>			
...	...	...	...

MSSPControlReq Message Format

Control Tag	Description
TAG_IP_ADDRESS	An IP Address. The IP address data for this tag is in the same format as in other messages, that is: AddressType USHORT 2 bytes, AddressLength USHORT 2 bytes, IPAddress BYTE[16] 16 bytes.
TAG_HOSTNAME	A hostname. The data for this tag is a variable-length character string (VARCHAR data type).
TAG_RADIUS_ATTR_ID	A RADIUS protocol Attribute ID. The data for this tag is in the format: AttributeID BYTE 1 bytes, AlignFiller BYTE[3] 3 bytes.
TAG_RADIUS_ATTR_VALUE	A RADIUS protocol attribute value. The data for this tag is in the format specified by the RADIUS protocol.

Control Tags

MSSPControlConf

Description: This message is sent by the MSSP 22 to acknowledge successful execution of a previous MSSPControlReq message.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_CONTROL_CONF.	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4
TakeControl	A value indicating whether the flow remains suspended at a trigger awaiting further control (MSSP_TRIGGER) or not (MSSP_EVENT_REPORT).	UINT	4
[ControlTag1]	One of the values identifying the associated control data field.	USHORT	2
[ControlLength1]	Length of following data, in	USHORT	2

Field Name	Description	Data Type	Byte Size
	bytes.		
[ControlData1]	The data field value.	*	*
[ControlTag2]			
...	...	...	...

MSSPControlConf Message Format

**MSSPReleaseReq**

Description: This message is issued to terminate an active flow. The flow to be terminated may be suspended at a trigger or may be active. The MSSP 22 will terminate the flow and provide any events or metering messages following the transmission of an *MSSPReleaseConf* message. The confirmation will be ordered prior to any events or metering messages resulting from the termination of the flow. If this message is being sent as the response to a trigger the suspended packet and its associated context is identified by the ControlID field within the request message. If the ControlID value in the message is zero then the (active) flow to be terminated is identified by the FlowID field.

The ReasonCode field will contain a numeric value indicating the reason for terminating the flow. The ReasonCode value will be stored in any detail records generated for the flow. An *MSSPReleaseConf* message will be sent to positively confirm a release flow operation. An *MSSPFailureConf* message with an error code of MSSP\_E\_INVALID\_CONTROL\_ID will be returned in the event the ControlID field is invalid. An *MSSPFailureConf* message with an error code of MSSP\_E\_INVALID\_FLOW\_ID will be returned in the event the FlowID field is invalid.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP RELEASE REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
ControlID	The ControlID of the control dialog that was created by the corresponding IDP event if the message is being sent as the response to a trigger. ControlID must be set to zero to release an active flow.	UINT	4

<b>Field Name</b>	<b>Description</b>	<b>Data Type</b>	<b>Byte Size</b>
FlowID	The identifier of the active flow to be released.	UINT	4
ReasonCode	A numeric code indicating the reason the connection was dropped. This code is stored in any flow detail records that are generated for the flow.	UINT	2

MSSPReleaseReq Message Format

### MSSPReleaseConf

Description: This message is sent by the MSSP 22 to acknowledge successful execution of a previous *MSSPReleaseReq* message.

Message Format:

<b>Field Name</b>	<b>Description</b>	<b>Data Type</b>	<b>Byte Size</b>
MessageType	A value identifying the type and format of this message, <u>MSSP RELEASE CONF</u> .	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4

MSSPReleaseConf Message Format

### MSSPActivityTestReq

Description: This request may be used to check the status of a previously reported flow or session. An *MSSPActivityTestConf* will be returned if the specified flow is still valid (active). If the flow identified by FlowID is not valid an *MSSPFailureConf* message will be sent as a confirmation with a failure code of MSSP\_E\_INVALID\_FLOW\_ID.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_ACTIVITY_TEST_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
FlowID	The identifier of the flow to be tested.	UINT	4

MSSPActivityTestReq Message Format

### MSSPActivityTestConf

Description: This message is sent by the MSSP 22 to acknowledge successful execution of a previous *MSSPActivityTestReq* message.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_ACTIVITY_TEST_CONF.	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4

MSSPActivityTestConf Message Format

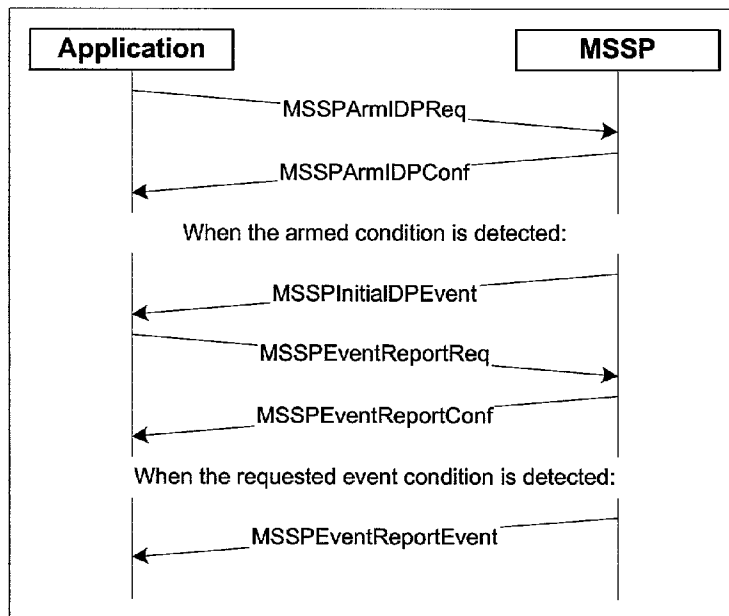
**Event Reporting Feature**

Description: The messages in this section allow an application to request additional event reports subsequent to an Initial Detection Point event. When the IDP that initiates a control dialog is a trigger, the application typically requests additional event reports via the *EventReportMask* and/or *TriggerMask* fields in an *MSSPContinueReq* or *MSSPConnectReq* response to the IDP event. When the IDP is not a trigger (ie, event report only without suspension of packet processing) the *MSSPEventReportReq* request is the only means of requesting additional event reports.

Privileges Required: EDP.

List of Messages: MSSPEventReportReq, MSSPEventReportConf, MSSPEventReportEvent.

Message Flow Diagram:



**MSSPEventReportReq**

Description: This message may be used to arm an event reporting detection point. The detection point will be armed as an event detection point for the indicated flow only, and events will be sent for cases where the flow passes through a control state specified within any of the event report or trigger masks. Upon receiving the event report request the MSSP 22 will arm the detection point(s) and send a confirmation indicating the success of the arming operation. In the event that a failure occurs while trying to arm the requested detection point(s) an *MSSPFailureConf* will be returned indicating the reason for the failure.

This request will result in *MSSPEventReportEvent* messages being sent to indicate that the flow has transitioned through the specified states. This request differs from the *MSSPArmIDPReq* in the sense that it is used to change event reporting for a

single, existing flow while the *MSSPArmIDPReq* request establishes the starting point where flows are first monitored.

All event reporting detection points remaining after a flow terminates are automatically disarmed by the MSSP 22. An *MSSPEventReportReq* for a flow that already has event reporting detection points armed has the effect of superceding the previous request. An *MSSPEventReportReq* with no *EventReportMask* or *TriggerMask* values set may be used to cancel all previously requested event reports and triggers for that control dialog.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_EVENT_REPORT_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
ControlID	The ControlID of the control dialog that was created by the corresponding IDP event.	UINT	4
PACKETDATACOPYBYTES	The number of bytes to be copied from the packet at the requested event report detection point. The value 0xFFFFFFFF may be used to request a copy of the entire IP packet (not including the media-specific layer2 header).	UINT	4

Field Name	Description	Data Type	Byte Size
EventReportMask	A bitwise combination of values from the appropriate detection point class table identifying the specific detection point(s) that should be armed for event reporting during this control dialog.	UINT	4
TriggerMask	A bitwise combination of values from the appropriate detection point class table identifying the specific detection point(s) that should be armed as a trigger during this control dialog.	UINT	4

MSSPEventReportReq Message Format

**MSSPEventReportConf**

Description: This message is sent by the MSSP 22 to acknowledge successful execution of a previous *MSSPEventReportReq* message.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_EVENT_REPORT_CONF.	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4

MSSPActivityTestConf Message Format

## EventReportEvent

Description: This message is sent by the MSSP 22 to acknowledge successful execution of a previous *MSSPActivityTestReq* message.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, <i>MSSP_EVENT_REPORT_EVENT</i> .	UINT	4
DETECTIONPOINTCLASS	One of the values identifying the detection point class that contains the detection point that caused this notification.	UINT	4
DetectionPoint	One of the values from the appropriate detection point class table identifying the specific detection point that caused this notification.	UINT	4
ResourceID	The MSSP-determined value that uniquely identifies the detection point criteria set that was matched. This is the same value that was returned in the <i>MSSPArmIDPConf</i> message that created the control dialog.	UINT	4
ControlID	The identifier of the control dialog that requested this event report.	UINT	4
TakeControl	A value indicating whether the flow is suspended at a trigger awaiting further control ( <i>MSSP_TRIGGER</i> ) or not ( <i>MSSP_EVENT_REPORT</i> ).	UINT	4
OPERATORID	Operator (when applicable to this detection point).	UINT	4
SUBSCRIBERGROUP ID	SubscriberGroup (when applicable to this detection point).	UINT	4
SessionID	SessionID (when applicable	UINT	4

A-44

Field Name	Description	Data Type	Byte Size
	to this detection point).		
FlowID	An MSSP-determined value that uniquely identifies the logical data path through the control logic that encountered the detection point.	UINT	4
SourceIPType	One of the symbolic constants: IP_NONE, IP_V4, IP_V6	USHORT	2
SourceIPLength	Length of SourceIPAddress (bits).	USHORT	2
SourceIPAddress	Source IPAddress (when applicable to this detection point).	BYTE[16]	16
SourcePort	Source port (when applicable to this detection point).	UINT	4
DestinationIPType	One of the symbolic constants: IP_NONE, IP_V4, IP_V6	USHORT	2
DestinationIPLength	Length of DestinationIPAddress (bits).	USHORT	2
DestinationIPAddress	Destination IPAddress (when applicable to this detection point).	BYTE[16]	16
DestinationPort	Destination port (when applicable to this detection point).	UINT	4
NumericValue1	A numeric value (IP protocol number, counter value, etc.), when applicable to this detection point.	UINT	4
NumericValue2	A numeric value (IP protocol number, counter value, etc.), when applicable to this detection point.	UINT	4
STRINGVALUE	A string value criteria (hostname, username, etc.), when applicable to this detection point.	VARCHAR	*
PacketData	Binary data extracted from the packet (if requested).	VARBYTE	*

MSSPEventReportEvent Message Format

A-45

**Service Filter Feature**

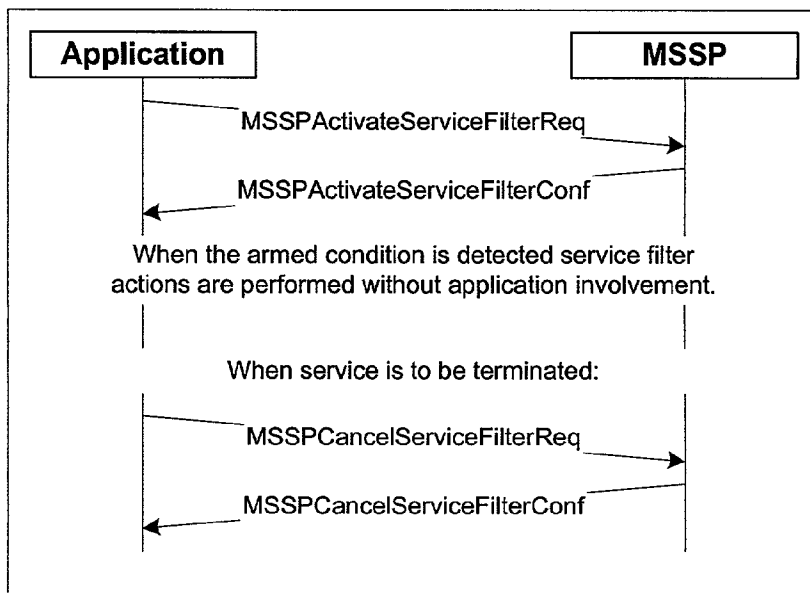
Description: The messages in this section allow an application to specify programmed behavior at a detection point that does not require the involvement of the application. The SourceIPAddress, SourcePort, DestinationIPAddress, and IPProtocolNumber fields will be used to match incoming requests to determine if the predefined service interaction should be executed. The matching process is similar to the process used for initial detection points in general and wildcards may be used in the fields to be matched.

If a flow matches the criteria, the actions specified in the remainder of the message will be carried out with no application involvement. Actions that may be specified include the reporting of events as well as redirecting a request to a specified redirection address and port number. The *EventReportMask* will be used to determine which events will be reported in the future for the flow if event reports are required. Criteria to be matched may not overlap with armed initial detection point criteria. If the request cannot be completed for any reason an *MSSPFailureConf* message will be returned with a matching RequestID and an error code indicating the nature of the failure. If the request completes successfully, an *MSSPActivateServiceFilterConf* is returned. Service Filtering will remain active until cancelled by an *MSSPCancelServiceFilterReq* request.

Privileges Required: ServiceFilter.

List of Messages: MSSPActivateServiceFilterReq, MSSPActivateServiceFilterConf, MSSPCancelServiceFilterReq, MSSPCancelServiceFilterConf.

Message Flow Diagrams:



### MSSPActivateServiceFilterReq

Description: This request is used to identify an initial detection point and specify the traffic criteria that should cause a pre-determined behavior to be applied.

When a flow encounters an initial detection point with a service filter and conditions match the service filter criteria, the pre-determined action in the service filter is applied to the packet and packet processing proceeds as directed. The criteria strings may contain wildcard characters that may be used to specify a wide range of triggers. The MSSP 22 sends an *MSSPActivateServiceFilterConf* message after the IDP is successfully armed with the Service Filter. In the event that error conditions prevent arming the IDP an *MSSPFailureConf* message is returned instead indicating the reason for the failure.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_ACTIVATE_SERVICE_FILTER_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
ServiceID	The identifier of the MSSP 22 service that this service filter is to be associated with.	UINT	4
DetectionPointClass	One of the values from identifying the detection point class that contains the detection point being armed.	UINT	4
DetectionPoint	One of the values from the appropriate detection point class table specifying the specific detection point to be armed. A service filter can only be armed at an IDP and trigger-capable detection point.	UINT	4
OperatorID	Operator criteria. The value 0xFFFFFFFF is a wild card that matches any OperatorID.	UINT	4
SubscriberGroupID	SubscriberGroup criteria. The value 0xFFFFFFFF is a wild card that matches any SubscriberGroupID.	UINT	4

Field Name	Description	Data Type	Byte Size
SessionID	SessionID criteria. The value 0xFFFFFFFF is a wild card that matches any SessionID.	UINT	4
SourceIPType	One of the symbolic constants: IP_NONE, IP_V4, IP_V6	USHORT	2
SourceIPLength	Length of SourceIPAddress (in bits) to be matched. For example, providing the IP_V4 address 151.104.0.0 with a length of 16 would match all IP addresses beginning with 151.104. A length of zero specifies a wildcard that matches any address.	USHORT	2
SourceIPAddress	Source IPAddress criteria.	BYTE[16]	16
SourcePort	Source port criteria. The value 0xFFFFFFFF is a wild card that matches any port number.	UINT	4
DestinationIPType	One of the symbolic constants: IP_NONE, IP_V4, IP_V6	USHORT	2
DestinationIPLength	Length of DestinationIPAddress (in bits). Wild cards may be specified as described for SourceIPLength, above.	USHORT	2
DestinationIPAddress	Destination IPAddress criteria.	BYTE[16]	16
DestinationPort	Destination port criteria. The value 0xFFFFFFFF is a wild card that matches any port number.	UINT	4
NumericValue1	A numeric value criteria (IP protocol number, counter value, etc.)	UINT	4
NumericValue2	A numeric value criteria (IP protocol number, counter value, etc.)	UINT	4
StringValue	A string value criteria (hostname, username, etc.).	VARCHAR	*
ActionCode	One of the following values indicating the type of service filter action to be performed: MSSP_ACTION_RELEASE, MSSP_ACTION_CONTINUE, or MSSP_ACTION_CONNECT.	UINT	4
ControlFlags	A bitwise combination of Control Flag values from on page 32.	UINT	4
ConnectIPType	One of the symbolic constants:	USHORT	2

Field Name	Description	Data type	Byte Size
	IP_NONE, IP_V4, IP_V6		
ConnectIPLength	Length of ConnectIPAddress (bits).	USHORT	2
ConnectIPAddress	IP address (for MSSP_ACTION_CONNECT).	BYTE[16]	16
ConnectPort	Port number (for MSSP_ACTION_CONNECT).	UINT	4
VPNID	The VPN or TunnelID (for MSSP_ACTION_CONNECT).	UINT	4
PACKETDATACOPY BYTES	The number of bytes to be copied from the packet at the requested event report detection point. The value 0xFFFFFFFF may be used to request a copy of the entire IP packet (not including the media-specific layer2 header).	UINT	4
EventReportMask	A bitwise combination of values from the appropriate detection point class table identifying the specific detection point(s) that should be armed for event reporting during this control dialog. Requires EDP privileges in addition to ServiceFilter privileges.	UINT	4

MSSPActivateServiceFilterReq Message Format

### MSSPActivateServiceFilterConf

Description: This message is sent by the MSSP 22 to acknowledge successful arming of a Service Filter Initial Detection Point by a previous *MSSPActivateServiceFilterReq* message.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_ACTIVATE_SERVICE_FILTER_CONF.	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4
ResourceID	An MSSP 22 determined value that uniquely identifies the newly established service filter detection point and arming criteria set. This value is required in order to cancel this service filter.	UINT	4

MSSPActivateServiceFilterConf Message Format

### MSSPCancelServiceFilterReq

Description: This request is used to cancel a service filter previously established by an *MSSPActivateServiceFilterReq* request.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_CANCEL_SERVICE_FILTER_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
ResourceID	The MSSP 22 determined value that uniquely identifies the service filter detection point and arming criteria set to be cancelled, as returned in the <i>MSSPActivateServiceFilterConf</i> message that created the service filter.	UINT	4

MSSPCancelServiceFilterReq Message Format

**MSSPCancelServiceFilterConf**

Description: This message is sent by the MSSP 22 to acknowledge successful cancellation of a Service Filter by a previous *MSSPCancelServiceFilterReq* message.

Message Format:

<b>Field Name</b>	<b>Description</b>	<b>Data Type</b>	<b>Byte Size</b>
MessageType	A value identifying the type and format of this message, MSSP_CANCEL_SERVICE_FILTER_CONF.	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4

MSSPCancelServiceFilterConf Message Format

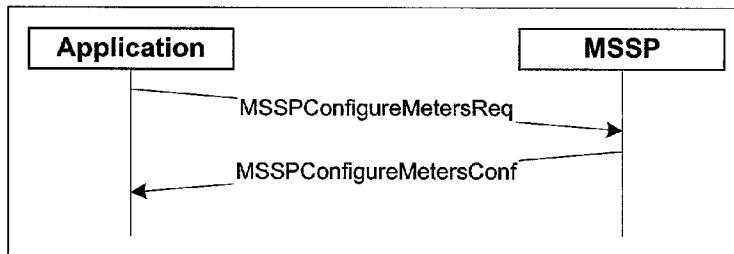
### Meter Configuration Feature

Description: The messages in this section allow an application to configure the data elements that will be metered by the MSSP 22. The meter configuration affects the meter elements that are populated in the *MSSPGetStatsConf* and *MSSPPeriodicStatsEvent* messages as well as detail records stored in the MSSP 22 database.

Privileges Required: MeterConfiguration.

List of Messages: MSSPConfigureMetersReq, MSSPConfigureMetersConf.

Message Flow Diagrams:



#### MSSPConfigureMetersReq

Description: This message is used to configure the metering performed by the MSSP 22 at the lowest level. The MeterClass field will contain one of two values, MSSP\_METER\_CLASS\_SESSION or MSSP\_METER\_CLASS\_FLOW. The class field will be used to indicate the scope of the metering request. The ObjectID field will identify instance of the object to be metered based on the MeterClass. For instance, if the MeterClass is MSSP\_METER\_CLASS\_SESSION the ObjectID will represent the session identifier, and if the MeteringType is MSSP\_METER\_CLASS\_FLOW the ObjectID will represent the flow identifier. The MetersEnabled field will contain a bit mask identifying the particular meters to be enabled within the class specified by the MeterClass field.

When the *MSSPConfigureMetersReq* request is sent for an object that already has meters configured, the MetersEnabled field specifies the new meter configuration for the object. A zero value in the mask position of a previously configured meter causes that meter to be disabled, and a non-zero value in the mask position of a previously unconfigured meter causes that meter to be enabled. The meter configuration affects the meter elements that are populated in the *MSSPGetStatsConf* and *MSSPPeriodicStatsEvent* messages as well as detail records stored in the MSSP 22 database.

The MSSP 22 will process the request and return an *MSSPConfigureMetersConf* message as a positive acknowledgement if the request is successful. An unsuccessful request will cause an *MSSPFailureConf* message to be sent as a negative acknowledgement. The failure code value will contain one of the following values depending on the request parameter that failed validation:

MSSP\_E\_INVALID\_METER\_CLASS,  
 MSSP\_E\_INVALID\_FLOW\_ID,  
 MSSP\_E\_INVALID\_SESSION\_ID, or  
 MSSP\_E\_INVALID\_FLOW\_METER\_MASK.

A-52

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_CONFIGURE_METERS_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
MeterClass	Defines the class of the object(s) to be metered: MSSP_METER_CLASS_SESSION, or MSSP_METER_CLASS_FLOW.	UINT	4
ObjectID	An identifier that uniquely identifies the flow or session object(s) to be metered.	UINT	4
EnabledMeterMask	A bitwise combination of one or more of the meter masks representing metering elements to be enabled.	UINT	4

MSSPConfigureMetersReq Message Format

Meter Element Mask	Description
MSSP_METER_BYTES_UPLINK	Number of uplink bytes transferred (includes all headers).
MSSP_METER_BYTES_DOWNLINK	Number of downlink bytes transferred (includes all headers).
MSSP_METER_BYTE_RATE_UPLINK	Uplink byte rate for this flow since last meter report
MSSP_METER_BYTE_RATE_DOWNLINK	Downlink byte rate for this flow since last meter report
MSSP_METER_PACKETS_UPLINK	Number of uplink packets transferred
MSSP_METER_PACKETS_DOWNLINK	Number of downlink packets transferred
MSSP_METER_IP_FRAGMENTS_UPLINK	Number of fragmented uplink IP packets
MSSP_METER_IP_FRAGMENTS_DOWNLINK	Number of fragmented

Meter Element Mask	Description
	downlink IP packets
MSSP_METER_IP_OPTIONS_UPLINK	Number of uplink IP packets with options set
MSSP_METER_IP_OPTIONS_DOWNLINK	Number of downlink IP packets with options set
MSSP_METER_TCP_BYTES_UPLINK	Number of uplink TCP bytes
MSSP_METER_TCP_BYTES_DOWNLINK	Number of downlink TCP bytes
MSSP_METER_TCP_PACKETS_UPLINK	Number of uplink TCP packets
MSSP_METER_TCP_PACKETS_DOWNLINK	Number of downlink TCP packets
MSSP_METER_TCP_PAYLOAD_UPLINK	Uplink TCP bytes less TCP header
MSSP_METER_TCP_PAYLOAD_DOWNLINK	Downlink TCP bytes less TCP header
MSSP_METER_TCP_NET_BYTES_UPLINK	Net uplink application bytes (differs from payload bytes due to adjustment for retransmissions)
MSSP_METER_TCP_NET_BYTES_DOWNLINK	Net downlink application bytes (differs from payload bytes due to adjustment for retransmissions)
MSSP_METER_UDP_BYTES_UPLINK	Number of uplink UDP bytes
MSSP_METER_UDP_BYTES_DOWNLINK	Number of downlink UDP bytes
MSSP_METER_UDP_PACKETS_UPLINK	Number of uplink UDP packets
MSSP_METER_UDP_PACKETS_DOWNLINK	Number of downlink UDP packets
MSSP_METER_ICMP_BYTES_UPLINK	Number of uplink ICMP bytes
MSSP_METER_ICMP_BYTES_DOWNLINK	Number of downlink ICMP bytes
MSSP_METER_ICMP_PACKETS_UPLINK	Number of uplink ICMP packets
MSSP_METER_ICMP_PACKETS_DOWNLINK	Number of downlink ICMP packets

Meter Element Masks

A-54

**MSSPConfigureMetersConf**

Description: This message is sent by the MSSP 22 to acknowledge successful execution of a previous *MSSPConfigureMetersReq* message.

Message Format:

<b>Field Name</b>	<b>Description</b>	<b>Data Type</b>	<b>Byte Size</b>
MessageType	A value identifying the type and format of this message, MSSP_CONFIGURE_METERS_CONF.	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4

MSSPConfigureMetersConf Message Format

**Charge Notification Feature**

Description: The messages in this section allow an application to request byte based reporting. Reporting may be requested on a session or flow basis. Session based charge notification effectively causes the same charge notification criteria to be applied to all flows in the session.

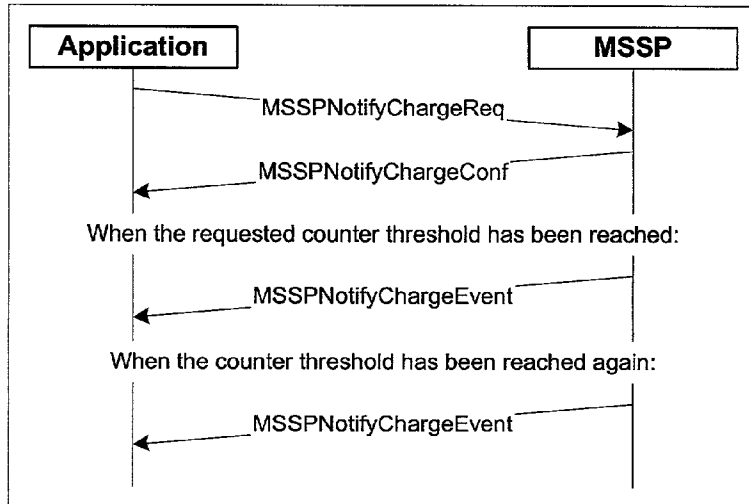
Registering for charge notification events will cause the number of bytes of the specified type transferred in uplink and downlink directions to be metered. Each time the reporting threshold is reached an *MSSPNotifyChargeEvent* message is sent from the MSSP 22 to the application indicating the number of bytes that have been transferred, and the counters are reset and begin counting towards the threshold again. Charge notification continues until the flow terminates or charge notification is explicitly cancelled by an *MSSPCancelNotifyChargeReq* request.

A packet is the atomic unit counted and each packet will either fall before the count is evaluated or after the count is evaluated. As a result, charge notification may not occur exactly on the byte count specified. For example, if notification was requested every 10K bytes, the notification may occur at 10.5 Kbytes if the packet that brought the count over 10K was slightly greater than 500 bytes. The actual counter values are provided in the *MSSPNotifyChargeEvent* message.

Privileges Required: ChargeNotification.

List of Messages: MSSPNotifyChargeReq, MSSPNotifyChargeConf, MSSPCancelNotifyChargeReq, MSSPCancelNotifyChargeConf, MSSPNotifyChargeEvent.

Message Flow Diagrams:



### MSSPNotifyChargeReq

Description: This request is used to register byte based reporting on either a session-wide or per-flow basis. An *MSSPNotifyChargeConf* message will be sent to indicate that charge notification has successfully been enabled. If charge notification cannot be enabled an *STL\_FAILURE\_CONF* message will be sent to indicate failure and the error code field will identify the reason for the failure.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_NOTIFY_CHARGE_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
Scope	Defines the scope of the object(s) to be metered: MSSP_SCOPE_SESSION, or MSSP_SCOPE_FLOW.	UINT	4
ObjectID	An identifier that uniquely identifies the flow or session object to be metered.	UINT	4
MeterType	One of the meter masks representing metering elements to be enabled. Regardless of whether the mask specified is an uplink or downlink meter mask, the corresponding uplink meter is compared to the UplinkThreshold value and the corresponding downlink meter is compared to the DownlinkThreshold value.	UINT	4
	Alignment padding for ULONG		4

Field Name	Description	Data Type	Byte Size
UplinkThreshold	The threshold to compare against the uplink Meter Type value when deciding if an event should be sent.	ULONG	8
DownlinkThreshold	The threshold to compare against the downlink MeterType value when deciding if an event should be sent.	ULONG	8
TotalThreshold	The threshold to compare against the total uplink+downlink meter value when deciding if an event should be sent.	ULONG	8

MSSPNotifyChargeReq Message Format

### MSSPNotifyChargeConf

Description: This message is sent by the MSSP 22 to acknowledge successful execution of a previous *MSSPNotifyChargeReq* message.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_NOTIFY_CHARGE_CONF.	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4
ResourceID	The MSSP-determined value that uniquely identifies the charge notification criteria set. This value is needed in order to cancel change notification.	UINT	4

MSSPNotifyChargeConf Message Format

**MSSPCancelNotifyChargeReq**

Description: This request is used to cancel byte based reporting established by a previous *MSSPNotifyChargeReq* request.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_CANCEL_NOTIFY_CHARGE_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
ResourceID	The MSSP-determined value uniquely identifying the charge notification criteria set to be cancelled, as returned in the <i>MSSPNotifyChargeConf</i> .	UINT	4

MSSPCancelNotifyChargeReq Message Format

**MSSPCancelNotifyChargeConf**

Description: This message is sent by the MSSP 22 to acknowledge successful execution of a previous *MSSPCancelNotifyChargeReq* message.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_CANCEL_NOTIFY_CHARGE_CONF.	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4

MSSPCancelNotifyChargeConf Message Format

### MSSPNotifyChargeEvent

Description: This message is used to notify the application that a previously registered charge notification threshold has been exceeded.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_NOTIFY_CHARGE_EVENT.	UINT	4
Scope	Defines the scope of the object being metered: MSSP_SCOPE_SESSION, or MSSP_SCOPE_FLOW.	UINT	4
ObjectID	An identifier that uniquely identifies the flow or session object being metered.	UINT	4
ResourceID	The MSSP-determined value that uniquely identifies the charge notification criteria set.	UINT	4
MeterType	The meter mask specified in the Notify Charge Request, identifying the metering elements used to generate this charge notification. The mask may be either an uplink or downlink meter mask.	UINT	4
	Alignment padding for ULONG		4
UplinkBytes	The uplink meter value at the time this message was sent.	ULONG	8
DownlinkBytes	The downlink meter value at the time this message was sent.	ULONG	8
TotalBytes	The total of the uplink and downlink meter values at the time this message was sent.	ULONG	8

MSSPNotifyChargeEvent Message Format

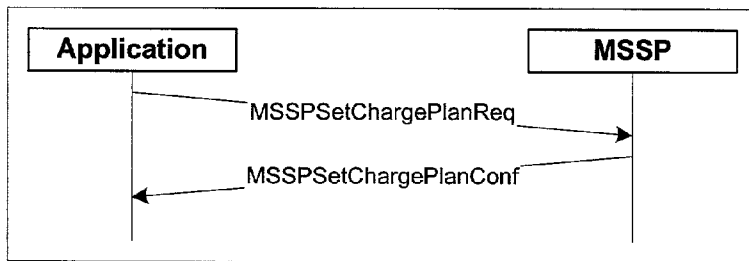
**Charge Plan Feature**

Description: The messages in this section allow an application to indicate the cost of the services provided and record the charge plan used in the MSSP 22 detail record.

Privileges Required: ChargePlan.

List of Messages: MSSPSetChargePlanReq, MSSPSetChargePlanConf.

Message Flow Diagram:



**MSSPSetChargePlanReq**

Description: This message is used to record the charge plan used for a service in the MSSP 22 detail record.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_SET_CHARGE_PLAN_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
FlowID	The identifier of the flow that is the subject of this request.	UINT	4
ChargePlanID	A value indicating the charge plan to be used for billing this service. This value is not interpreted by the MSSP 22 and is stored in the detail record unchanged.	UINT	4

MSSPSetChargePlanReq Message Format

**MSSPSetChargePlanConf**

Description: This message is sent by the MSSP 22 to acknowledge successful execution of a previous *MSSPSetChargePlanReq* message.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_SET_CHARGE_PLAN_CONF.	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4

MSSPSetChargePlanConf Message Format

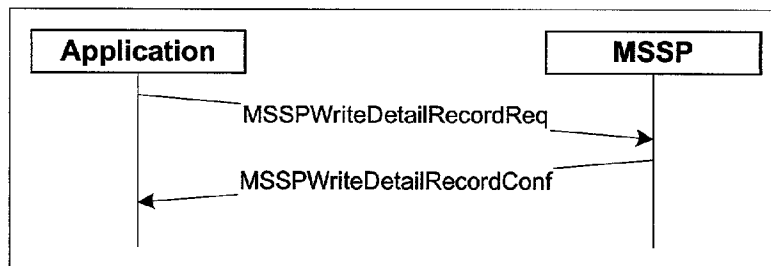
**Detail Record Control Feature**

Description: The messages in this section allow an application to control when MSSP 22 detail records are written.

Privileges Required: DetailRecordControl.

List of Messages: MSSPWriteDetailRecordReq, MSSPWriteDetailRecordConf.

Message Flow Diagrams:



### MSSPWriteDetailRecordReq

Description: This message allows an application to control when detail records are written to the MSSP 22 database. Detail records that are written by this request automatically have a ReasonCode of MSSP\_RC\_PARTIAL\_DETAIL assigned. Partial detail records are commonly used to guarantee that, in the event of an unrecoverable error, all but the most recent activities of a subscriber interaction are captured for billing purposes.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_WRITE_DETAIL_RECORD_REQ.	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
FlowID	The identifier of the flow that is the subject of this request.	UINT	4

MSSPWriteDetailRecordReq Message Format

### MSSPWriteDetailRecordConf

Description: This message is sent by the MSSP 22 to acknowledge successful execution of a previous *MSSPWriteDetailRecordReq* message.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_WRITE_DETAIL_RECORD_CONF.	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4

MSSPWriteDetailRecordConf Message Format

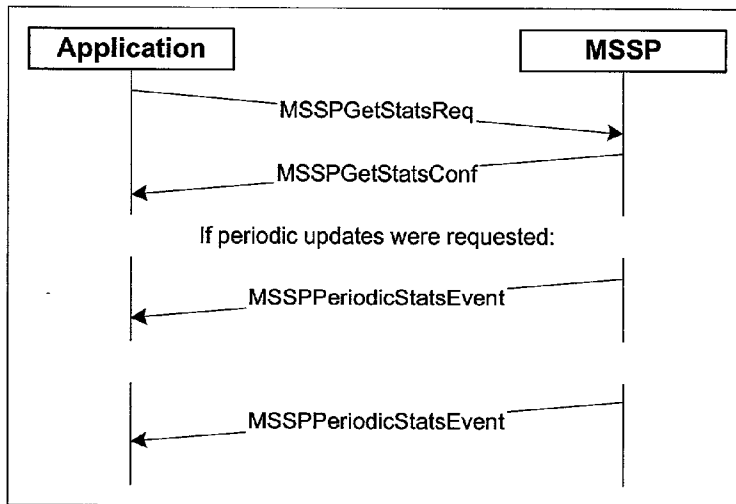
**Statistics Feature**

Description: The messages in this section allow an application to obtain various statistics for a session or flow managed by the application.

Privileges Required: Statistics.

List of Messages: MSSPGetStatsReq, MSSPGetStatsConf, MSSPPeriodicStatsEvent.

Message Flow Diagrams:



**MSSPGetStatsReq**

Description: This request is used to request session or flow statistics. In addition to the current values of the statistics the request may optionally request future updates, either periodically or upon flow or session termination. Statistical values depend upon the meters configured by previous *MSSPConfigureMetersReq* requests.

The MSSP 22 will process the request and return an *MSSPGetStatsConf* message with the current statistical values as a positive acknowledgement if the request is successful. In addition, *MSSPPeriodicStatsEvent* messages will be sent if future updates were requested via the *Interval* field. An unsuccessful request will cause an *MSSPFailureConf* message to be sent as a negative acknowledgement. The failure code value will contain one of the following values depending on the request parameter that failed validation:

- MSSP\_E\_INVALID\_STATS\_TYPE,
- MSSP\_E\_INVALID\_FLOW\_ID,
- MSSP\_E\_INVALID\_SESSION\_ID, or
- MSSP\_E\_INVALID\_INTERVAL.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, <code>MSSP_GET_STATS_REQ</code> .	UINT	4
RequestID	Any value, to be returned in the corresponding response message. Intended to uniquely identify request/response pairs when multiple requests are made concurrently.	UINT	4
Interval	The desired interval (in seconds) between future statistics updates. Set to 0 when no future updates are requested, or <code>0xFFFF FFFF</code> to request a single update at the termination of the flow or session. Intervals must be at least 30 seconds.	UINT	4
StatsType	Defines the type of statistics being requested: <code>MSSP_STATS_TYPE_SESSION</code> , or <code>MSSP_STATS_TYPE_FLOW</code> .	UINT	4
ObjectID	An identifier that uniquely identifies the flow or session whose statistics are desired.	UINT	4

MSSPGetStatsReq Message Format

**MSSPGetStatsConf**

Description: This request is used to return the session or flow statistics requested by a previous *MSSPGetStatsReq* request. Statistical values depend upon the meters configured by previous *MSSPConfigureMetersReq* requests.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, <code>MSSP_GET_STATS_CONF</code> .	UINT	4
RequestID	The RequestID value from the corresponding request message.	UINT	4
StatsType	Defines the type of statistics that are present in this	UINT	4

Field Name	Description	Data Type	Byte Size
	message: MSSP_STATS_TYPE_SESSION, or MSSP_STATS_TYPE_FLOW.		
ObjectID	An identifier that uniquely identifies the flow or session associated with the statistics.	UINT	4
ServiceCardSlot	The chassis slot number of the Service Card handling this flow or session.	UINT	4
Termination	A non-zero value indicates that this is the final meter indication for this flow or session.	UINT	2
ReasonCode	When Termination is non-zero, this field contains a numeric value indicating the reason for termination of the flow or session.	UINT	2
Duration	The time duration over which the reported statistics were collected, in seconds.	UINT	4
EnabledMeterMask	A bitwise combination of meter masks indicating the metering elements that are configured.	UINT	4
BytesUplink	* Total uplink bytes (including all headers).	ULONG	8
BytesDownlink	* Total downlink bytes (including all headers).	ULONG	8
ByteRateUplink	* The uplink byte rate per second.	ULONG	8
ByteRateDownlink	* The downlink byte rate per second.	ULONG	8
PacketsUplink	* The number of uplink packets transferred.	ULONG	8
PacketsDownlink	* The number of downlink packets transferred.	ULONG	8
IPFragmentsUplink	* The number of fragmented uplink IP packets.	ULONG	8
IPFragmentsDownlink	* The number of fragmented downlink IP packets.	ULONG	8
IPOptionsUplink	* The number of uplink IP packets with options set.	ULONG	8
IPOptionsDownlink	* The number of downlink IP	ULONG	8

Field Name	Description	Data Type	Byte Size
	packets with options set.		
TCPBytesUplink	* The number of uplink TCP bytes.	ULONG	8
TCPBytesDownlink	* The number of downlink TCP bytes.	ULONG	8
TCPPackagesUplink	* The number of uplink TCP packets.	ULONG	8
TCPPackagesDownlink	* The number of downlink TCP packets.	ULONG	8
TCPPayloadUplink	* The number of uplink TCP Gross payload bytes.	ULONG	8
TCPPayloadDownlink	* The number of downlink TCP Gross payload * bytes.	ULONG	8
TCPNetBytesUplink	* The number of uplink TCP Net payload bytes.	ULONG	8
TCPNetBytesDownlink	* The number of downlink TCP Net payload bytes.	ULONG	8
UDPBytesUplink	* The number of uplink UDP bytes.	ULONG	8
UDPBytesDownlink	* The number of downlink UDP bytes.	ULONG	8
UDPPackagesUplink	* The number of uplink UDP packets.	ULONG	8
UDPPackagesDownlink	* The number of downlink UDP packets.	ULONG	8
ICMPBytesUplink	* The number of uplink ICMP bytes	ULONG	8
ICMPBytesDownlink	* The number of downlink ICMP bytes	ULONG	8
ICMPPackagesUplink	* The number of uplink ICMP packets.	ULONG	8
ICMPPackagesDownlink	* The number of downlink ICMP packets.	ULONG	8

MSSPGetStatsConf Message Format

Fields marked with \* contain valid data only when the corresponding meter is configured in the MSSP 22 (as indicated in the *EnabledMeterMask* field).

## MSSPPeriodicStatsEvent

Description: This request is used to return the periodic updates of session or flow statistics that were requested by a previous *MSSPGetStatsReq* request. Statistical values depend upon the meters configured by previous *MSSPConfigureMetersReq* requests.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_PERIODIC_STATS_EVENT.	UINT	4
StatsType	Defines the type of statistics that are present in this message: MSSP_STATS_TYPE_SESSION, or SSP_STATS_TYPE_FLOW.	UINT	4
ObjectID	An identifier that uniquely identifies the flow or session associated with the statistics.	UINT	4
ServiceCardSlot	The chassis slot number of the Service Card handling this flow or session.	UINT	4
Termination	A non-zero value indicates that this is the final meter indication for this flow or session.	UINT	2
ReasonCode	When Termination is non-zero, this field contains a numeric value indicating the reason for termination of the flow or session.	UINT	2
Duration	The time duration over which the reported statistics were collected, in seconds.	UINT	4
EnabledMeterMask	A bitwise combination of meter masks indicating the metering elements that are configured.	UINT	4
BytesUplink	* Total uplink bytes (including all headers).	ULONG	8
BytesDownlink	* Total downlink bytes (including all headers).	ULONG	8
ByteRateUplink	* The uplink byte rate per second.	ULONG	8
ByteRateDownlink	* The downlink byte rate per second.	ULONG	8
PacketsUplink	* The number of uplink packets transferred.	ULONG	8
PacketsDownlink	* The number of downlink packets	ULONG	8

Field Name	Description	Data Type	Byte Size
	transferred.		
IPFragmentsUplink	* The number of fragmented uplink IP packets.	ULONG	8
IPFragmentsDownlink	* The number of fragmented downlink IP packets.	ULONG	8
IPOptionsUplink	* The number of uplink IP packets with options set.	ULONG	8
IPOptionsDownlink	* The number of downlink IP packets with options set.	ULONG	8
TCPBytesUplink	* The number of uplink TCP bytes.	ULONG	8
TCPBytesDownlink	* The number of downlink TCP bytes.	ULONG	8
TCPPackagesUplink	* The number of uplink TCP packets.	ULONG	8
TCPPackagesDownlink	* The number of downlink TCP packets.	ULONG	8
TCPPayloadUplink	* The number of uplink TCP Gross payload bytes.	ULONG	8
TCPPayloadDownlink	* The number of downlink TCP Gross payload * bytes.	ULONG	8
TCPNetBytesUplink	* The number of uplink TCP Net payload bytes.	ULONG	8
TCPNetBytesDownlink	* The number of downlink TCP Net payload bytes.	ULONG	8
UDPBytesUplink	* The number of uplink UDP bytes.	ULONG	8
UDPBytesDownlink	* The number of downlink UDP bytes.	ULONG	8
UDPPackagesUplink	* The number of uplink UDP packets.	ULONG	8
UDPPackagesDownlink	* The number of downlink UDP packets.	ULONG	8
ICMPBytesUplink	* The number of uplink ICMP bytes	ULONG	8
ICMPBytesDownlink	* The number of downlink ICMP bytes	ULONG	8
ICMPPackagesUplink	* The number of uplink ICMP packets.	ULONG	8
ICMPPackagesDownlink	* The number of downlink ICMP packets.	ULONG	8

MSSPPeriodicStatsEvent Message Format

Fields marked with \* contain valid data only when the corresponding meter is configured in the MSSP 22 (as indicated in the *EnabledMeterMask* field).

**Application Monitor Feature**

Description: The messages in this section allow an application to monitor the status of other applications connected to the same MSSP 22 instance.

Privileges Required: ApplicationMonitor.

List of Messages: MSSPAppSessionEvent.

**MSSPAppSessionEvent**

Description: This message is sent by the MSSP 22 to report the occurrence of an application session event. This message is also sent by the MSSP 22 to an application with ApplicationMonitor privileges immediately after its session is opened, informing it of other (pre-existing) application sessions.

Message Format:

Field Name	Description	Data Type	Byte Size
MessageType	A value identifying the type and format of this message, MSSP_APP_SESSION_EVENT.	UINT	4
EventType	One of the following values indicating the type of application session event that is being reported: MSSP_APP_SESSION_OPENED, MSSP_APP_SESSION_CLOSED, MSSP_APP_SESSION_TERMINATED.	UINT	4
ApplicationID	The MSSP-determined value that uniquely identifies this application in the MSSP 22 configuration database.	UINT	4
ReasonCode	For Closed and Terminated events, a value indicating the reason.	UINT	4
MaxResources	The maximum number of MSSP 22 resources that the application is allowed to use. A value of 0xFFFF FFFF indicates no limit.	UINT	4
APIType	A value indicating the type of API the application is using to communicate with the MSSP 22. At present only MSSP_API_ is defined.	UINT	2
ConnectSide	One of the following values indicating which side of the MSSP 22 the application is or was connected to: MSSP_EVEN_SIDE,	UINT	2

Field Name	Description	Data Type	Byte Size
	MSSP_ODD_SIDE.		
ConnectTime	The MSSP 22 system time when the application session was connected.	TIME	4
FeatureMask	A bitwise combination of one or more of the feature masks corresponding to the features that the application is authorized to use.	UINT	4
AppIPType	One of the symbolic constants: IP_NONE, IP_V4, IP_V6	USHORT	2
AppIPLength	Length of AppIPAddress (bits).	USHORT	2
AppIPAddress	Application IP address.	BYTE[16]	16
AppPort	Application port.	UINT	4
AppName	Application name (may be up to 32 non-null characters plus a NULL string termination character, additional message field bytes added to preserve field alignment).	CHAR[36]	36

MSSPAppSessionEvent Message Format

What is claimed is:

1. A method comprising:
  - in a network, receiving messages from an application program in an application program interface (API); and
  - passing the messages from the API to a control process in a mobile service switching platform (MSSP).
2. The method of claim 1 in which the network is a wireless network.
3. The method of claim 2 in which the wireless network is a second generation wireless network.
4. The method of claim 2 in which the wireless network is a GSM network.
5. The method of claim 2 in which the wireless network is a GPRS-enabled GSM network.
6. The method of claim 2 in which the wireless network is a TDMA network.
7. The method of claim 2 in which the wireless network is a CDMA network.
8. The method of claim 2 in which the wireless network is a UMTS network.
9. The method of claim 2 in which the wireless network is a TETRA network.
10. The method of claim 2 in which the wireless network is a Tetrapol network.
11. The method of claim 2 in which the wireless network is a DECT network.
12. The method of claim 2 in which the wireless network is an AMPS network.
13. The method of claim 2 in which the wireless network is a WLAN.
14. The method of claim 2 in which the wireless network is a third generation wireless network.
15. The method of claim 1 in which the API provides a protocol that allows the application program to control switching and routing functions in the MS SP.
16. The method of claim 1 in which the API provides a protocol that allows the application program to redirect packet flow through the MSSP on a per-flow basis.
17. The method of claim 1 in which the API provides a protocol that allows the application program to control policy decisions within the MSSP.
18. The method of claim 1 in which the API provides a protocol that allows the application program to arm initial detection points (IDPs) and services associated IDP events in the control process.
19. The method of claim 1 in which the API provides a protocol that allows the application program to disarm IDPs and service associated ICP events in the control process.
20. The method of claim 1 in which the API provides a protocol that allows the application program to request event reports.
21. The method of claim 1 in which the API provides a protocol that allows the application program to specify programmed behavior at a detection point in the control process.
22. The method of claim 1 in which the API provides a protocol that allows the application program to configure data elements that are metered by the control process of the MSSP.
23. The method of claim 1 in which the API provides a protocol that allows the application program to request byte-based reporting.
  24. The method of claim 23 in which the reporting is session-based.
  25. The method of claim 23 in which the reporting is service interaction based.
  26. The method of claim 23 in which the reporting is flow-based.
  27. The method of claim 1 in which the API provides a protocol that allows the application program to specify a cost of services provided.
  28. The method of claim 27 in which the protocol allows the application program to record a charge plan used in a detail record.
  29. The method of claim 28 in which the protocol allows the application program to control when the detail record is written.
  30. The method of claim 1 in which the API provides a protocol that allows the application program to obtain statistics for a session managed by the application program.
  31. The method of claim 1 in which the API provides a protocol that allows the application program to obtain statistics for a flow managed by the application program.
  32. The method of claim 1 in which the API provides a protocol that allows the application program to monitor a status of other applications connected to the control process of the MSSP.
  33. An application program interface (API) comprising:
    - a set of application layer protocols that allows exchange of messages between an external application process and a control process residing in a Mobile Service Switching Platform (MSSP) using Transmission Control Protocol/Internet Protocol (TCP/IP) network services.
    34. The method of claim 33 in which the set comprises a protocol that allows the application process to control switching and routing functions in the MS SP.
    35. The method of claim 33 in which the set comprises a protocol that allows the application process to redirect packet flow through the MSSP on a per-flow basis.
    36. The method of claim 33 in which the set comprises a protocol that allows the application program to control policy decisions within the MSSP.
    37. The API of claim 33 in which the set of application layers protocols comprises a protocol that allows the application process to arm initial detection points (IDPs) and services associated IDP events in the control process.
    38. The API of claim 33 in which the set of application layers protocols comprises a protocol that allows the application process to disarm initial detection points (IDPs) and services associated IDP events in the control process.
    39. The API of claim 33 in which the set of application layers protocols comprises a protocol that allows the application process to request event reports from the control process.
    40. The API of claim 33 in which the set of application layers protocols comprises a protocol that allows the application process to specify programmed behavior at a detection point in the control process.
    41. The API of claim 33 in which the set of application layers protocols comprises a protocol that allows the application process to configure data elements that are metered by the control process.

**42.** The API of claim 33 in which the set of application layers protocols comprises a protocol that allows the application process to request byte-based reporting in the control process.

**43.** The API of claim 42 in which the reporting is session-based.

**44.** The API of claim 42 in which the reporting is service interaction-based.

**45.** The API of claim 42 in which the reporting is flow-based.

**46.** The API of claim 33 in which the set of application layers protocols comprises a protocol that allows the application process to specify a cost of services provided by the MSSP.

**47.** The API of claim 33 in which the set of application layers protocols comprises a protocol that allows the application process to record a charge plan used in a detail record stored in the MSSP.

**48.** The API of claim 33 in which the set of application layers protocols comprises a protocol that allows the application process to control when the detail record is written.

**49.** The API of claim 33 in which the set of application layers protocols comprises a protocol that allows the application process to obtain statistics for a session managed by the application process.

**50.** The API of claim 33 in which the set of application layers protocols comprises a protocol that allows the application process to obtain statistics for a flow managed by the application process.

**51.** The API of claim 33 in which the set of application layers protocols comprises a protocol that allows the application process to monitor a status of other application processes connected to the control process.

**52.** A system comprising:

a Gateway General Packet Radio Service Support Node (GGSN) linked to control process in a Mobile Service Switching Platform (MSSP);

a group of globally connected computers linked to the control process;

an application program interface (API) connected to the control process; and

an application system executing an application process linked to the API.

**53.** The system of claim 52 further comprising a General Packet Radio Service Support Node linked to the GGSN.

**54.** The system of claim 53 further comprising a Base Station Controller (BSC) linked to the General Packet Radio Service Support Node.

**55.** The system of claim 54 further comprising a Base Transceiver Station (BTS) linked to the BSC.

**56.** The system of claim 55 further comprising a mobile station (MS) linked to the BTS.

**57.** The system of claim 52 in which the API comprises a set of application layer protocols that allows exchange of messages between the application process and the control process.

**58.** The system of claim 57 in which the set of application layers protocols comprises a protocol that allows the application process to arm initial detection points (IDPs) and services associated IDP events in the control process.

**59.** The system of claim 57 in which the set of application layers protocols comprises a protocol that allows the application process to disarm initial detection points (IDPs) and services associated IDP events in the control process.

**60.** The system of claim 57 in which the set of application layers protocols comprises a protocol that allows the application process to request event reports from the control process.

**61.** The system of claim 57 in which the set of application layers protocols comprises a protocol that allows the application process to specify programmed behavior at a detection point in the control process.

**62.** The system of claim 57 in which the set of application layers protocols comprises a protocol that allows the application process to configure data elements that are metered by the control process.

**63.** The system of claim 57 in which the set of application layers protocols comprises a protocol that allows the application process to request byte-based reporting in the control process.

**64.** The API of claim 63 in which the reporting is session-based.

**65.** The API of claim 63 in which the reporting is flow-based.

**66.** The API of claim 63 in which the reporting is service interaction-based.

**67.** The system of claim 57 in which the set of application layers protocols comprises a protocol that allows the application process to specify a cost of services provided by the MSSP.

**68.** The system of claim 57 in which the set of application layers protocols comprises a protocol that allows the application process to record a charge plan used in a detail record stored in the MSSP.

**69.** The API of claim 68 in which the set of application layers protocols comprises a protocol that allows the application process to control when the detail record is written.

**70.** The system of claim 57 in which the set of application layers protocols comprises a protocol that allows the application process to obtain statistics for a session managed by the application process.

**71.** The system of claim 57 in which the set of application layers protocols comprises a protocol that allows the application process to obtain statistics for a flow managed by the application process.

**72.** The system of claim 57 in which the set of application layers protocols comprises a protocol that allows the application process to monitor a status of other application processes connected to the control process.

\* \* \* \* \*