```
{ atEntry }
```

and specifying, using a protocol-specific mechanism, the object instance

```
{ atNetAddress } = { internet "10.0.0.52" }
```

refers to all instances of entries in the table for which the associated atNetAddress value is { internet "10.0.0.52" }.

Each management protocol must provide a mechanism for accessing simple (non-aggregate) object types. Each management protocol specifies whether or not it supports access to aggregate object types. Further, the protocol must specify which instances are "returned" when an object type/instance pairing refers to more than one instance of a type.

To afford support for a variety of management protocols, all information by which instances of a given object type may be usefully distinguished, one from another, is represented by instances of object types defined in the MIB.

4.3. Macros for Managed Objects

In order to facilitate the use of tools for processing the definition of the MIB, the OBJECT-TYPE macro may be used. This macro permits the key aspects of an object type to be represented in a formal way.

```
OBJECT-TYPE MACRO ::=
BEGIN
    TYPE NOTATION ::= "SYNTAX" type (TYPE ObjectSyntax)
                      "ACCESS" Access
                      "STATUS" Status
    VALUE NOTATION ::= value (VALUE ObjectName)

    Access ::= "read-only"
                    | "read-write"
                    | "write-only"
                    | "not-accessible"
    Status ::= "mandatory"
                    | "optional"
                    | "obsolete"
    END
```

Given the object types defined earlier, we might imagine the following definitions being present in the MIB:

```
atIndex OBJECT-TYPE
```


SAP 1002 (Part 4 of 4)
CBM of U.S. Patent No. 8,037,158

```
                  SYNTAX  INTEGER
                  ACCESS  read-write
                  STATUS  mandatory
                  ::= { atEntry 1 }

          atPhysAddress OBJECT-TYPE
                  SYNTAX  OCTET STRING
                  ACCESS  read-write
                  STATUS  mandatory
                  ::= { atEntry 2 }

          atNetAddress OBJECT-TYPE
                  SYNTAX  NetworkAddress
                  ACCESS  read-write
                  STATUS  mandatory
                  ::= { atEntry 3 }

          atEntry OBJECT-TYPE
                  SYNTAX  AtEntry
                  ACCESS  read-write
                  STATUS  mandatory
                  ::= { atTable 1 }

          atTable OBJECT-TYPE
                  SYNTAX  SEQUENCE OF AtEntry
                  ACCESS  read-write
                  STATUS  mandatory
                  ::= { at 1 }

          AtEntry ::= SEQUENCE {
              atIndex
                  INTEGER,
              atPhysAddress
                  OCTET STRING,
              atNetAddress
                  NetworkAddress
          }
```

The first five definitions describe object types, relating, for example, the OBJECT DESCRIPTOR atIndex to the OBJECT IDENTIFIER { atEntry 1 }. In addition, the syntax of this object is defined (INTEGER) along with the access permitted (read-write) and status (mandatory). The sixth definition describes an ASN.1 type called AtEntry.

5. Extensions to the MIB

Every Internet-standard MIB document obsoletes all previous such documents. The portion of a name, termed the tail, following the OBJECT IDENTIFIER

{ mgmt version-number }

used to name objects shall remain unchanged between versions. New versions may:

(1) declare old object types obsolete (if necessary), but not delete their names;

(2) augment the definition of an object type corresponding to a list by appending non-aggregate object types to the object types in the list; or,

(3) define entirely new object types.

New versions may not:

(1) change the semantics of any previously defined object without changing the name of that object.

These rules are important because they admit easier support for multiple versions of the Internet-standard MIB. In particular, the semantics associated with the tail of a name remain constant throughout different versions of the MIB. Because multiple versions of the MIB may thus coincide in "tail-space," implementations supporting multiple versions of the MIB can be vastly simplified.

However, as a consequence, a management agent might return an instance corresponding to a superset of the expected object type. Following the principle of robustness, in this exceptional case, a manager should ignore any additional information beyond the definition of the expected object type. However, the robustness principle requires that one exercise care with respect to control actions: if an instance does not have the same syntax as its expected object type, then those control actions must fail. In both the monitoring and control cases, the name of an object returned by an operation must be identical to the name requested by an operation.

6. Definitions

```
RFC1155-SMI DEFINITIONS ::= BEGIN

EXPORTS -- EVERYTHING
        internet, directory, mgmt,
        experimental, private, enterprises,
        OBJECT-TYPE, ObjectName, ObjectSyntax, SimpleSyntax,
        ApplicationSyntax, NetworkAddress, IpAddress,
        Counter, Gauge, TimeTicks, Opaque;

 -- the path to the root

 internet      OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }

 directory     OBJECT IDENTIFIER ::= { internet 1 }

 mgmt          OBJECT IDENTIFIER ::= { internet 2 }

 experimental  OBJECT IDENTIFIER ::= { internet 3 }

 private       OBJECT IDENTIFIER ::= { internet 4 }
 enterprises   OBJECT IDENTIFIER ::= { private 1 }


 -- definition of object types

 OBJECT-TYPE MACRO ::=
 BEGIN
     TYPE NOTATION ::= "SYNTAX" type (TYPE ObjectSyntax)
                       "ACCESS" Access
                       "STATUS" Status
     VALUE NOTATION ::= value (VALUE ObjectName)

     Access ::= "read-only"
                     | "read-write"
                     | "write-only"
                     | "not-accessible"
     Status ::= "mandatory"
                     | "optional"
                     | "obsolete"
 END

    -- names of objects in the MIB

    ObjectName ::=
        OBJECT IDENTIFIER
```

```
-- syntax of objects in the MIB

ObjectSyntax ::=
    CHOICE {
        simple
            SimpleSyntax,

-- note that simple SEQUENCEs are not directly
-- mentioned here to keep things simple (i.e.,
-- prevent mis-use).  However, application-wide
-- types which are IMPLICITly encoded simple
-- SEQUENCEs may appear in the following CHOICE

        application-wide
            ApplicationSyntax
    }

   SimpleSyntax ::=
       CHOICE {
           number
               INTEGER,

           string
               OCTET STRING,

           object
               OBJECT IDENTIFIER,

           empty
               NULL
       }

   ApplicationSyntax ::=
       CHOICE {
           address
               NetworkAddress,

           counter
               Counter,

           gauge
               Gauge,

           ticks
               TimeTicks,

           arbitrary
               Opaque
```

```
-- other application-wide types, as they are
-- defined, will be added here
    }


-- application-wide types

NetworkAddress ::=
    CHOICE {
        internet
            IpAddress
    }

IpAddress ::=
    [APPLICATION 0]          -- in network-byte order
        IMPLICIT OCTET STRING (SIZE (4))

Counter ::=
    [APPLICATION 1]
        IMPLICIT INTEGER (0..4294967295)

Gauge ::=
    [APPLICATION 2]
        IMPLICIT INTEGER (0..4294967295)

TimeTicks ::=
    [APPLICATION 3]
        IMPLICIT INTEGER (0..4294967295)

Opaque ::=
    [APPLICATION 4]          -- arbitrary ASN.1 value,
        IMPLICIT OCTET STRING   --   "double-wrapped"

END
```

7. Acknowledgements

This memo was influenced by three sets of contributors to earlier drafts:

First, Lee Labarre of the MITRE Corporation, who as author of the NETMAN SMI [4], presented the basic roadmap for the SMI.

Second, several individuals who provided valuable comments on this memo prior to its initial distribution:

James R. Davin, Proteon
Mark S. Fedor, NYSERNet
Craig Partridge, BBN Laboratories
Martin Lee Schoffstall, Rensselaer Polytechnic Institute
Wengyik Yeong, NYSERNet

Third, the IETF MIB working group:

Karl Auerbach, Epilogue Technology
K. Ramesh Babu, Excelan
Lawrence Besaw, Hewlett-Packard
Jeffrey D. Case, University of Tennessee at Knoxville
James R. Davin, Proteon
Mark S. Fedor, NYSERNet
Robb Foster, BBN
Phill Gross, The MITRE Corporation
Bent Torp Jensen, Convergent Technology
Lee Labarre, The MITRE Corporation
Dan Lynch, Advanced Computing Environments
Keith McCloghrie, The Wollongong Group
Dave Mackie, 3Com/Bridge
Craig Partridge, BBN (chair)
Jim Robertson, 3Com/Bridge
Marshall T. Rose, The Wollongong Group
Greg Satz, cisco
Martin Lee Schoffstall, Rensselaer Polytechnic Institute
Lou Steinberg, IBM
Dean Throop, Data General
Unni Warrier, Unisys

8. References

[1] Information processing systems - Open Systems Interconnection, "Specification of Abstract Syntax Notation One (ASN.1)", International Organization for Standardization, International Standard 8824, December 1987.

[2] McCloghrie K., and M. Rose, "Management Information Base for Network Management of TCP/IP-based Internets", RFC 1156, Performance Systems International and Hughes LAN Systems, May 1990.

[3] Case, J., M. Fedor, M. Schoffstall, and J. Davin, The Simple Network Management Protocol", RFC 1157, University of Tennessee at Knoxville, Performance Systems International, Performance Systems International, and the MIT Laboratory for Computer Science, May 1990.

[4] LaBarre, L., "Structure and Identification of Management Information for the Internet", Internet Engineering Task Force working note, Network Information Center, SRI International, Menlo Park, California, April 1988.

[5] Cerf, V., "IAB Recommendations for the Development of Internet Network Management Standards", RFC 1052, IAB, April 1988.

[6] Cerf, V., "Report of the Second Ad Hoc Network Management Review Group", RFC 1109, IAB, August 1989.

[7] Information processing systems - Open Systems Interconnection, "Specification of Basic Encoding Rules for Abstract Notation One (ASN.1)", International Organization for Standardization, International Standard 8825, December 1987.

Security Considerations

Security issues are not discussed in this memo.

Authors' Addresses

Marshall T. Rose
PSI, Inc.
PSI California Office
P.O. Box 391776
Mountain View, CA 94039

Phone: (415) 961-3380

EMail: mrose@PSI.COM

Keith McCloghrie
The Wollongong Group
1129 San Antonio Road
Palo Alto, CA 04303

Phone: (415) 962-7160

EMail: sytek!kzm@HPLABS.HP.COM

Network Working Group
Request for Comments: 1157
Obsoletes: RFC 1098

J. Case
SNMP Research
M. Fedor
Performance Systems International
M. Schoffstall
Performance Systems International
J. Davin
MIT Laboratory for Computer Science
May 1990

A Simple Network Management Protocol (SNMP)

Table of Contents

1. Status of this Memo

This RFC is a re-release of RFC 1098, with a changed "Status of this Memo" section plus a few minor typographical corrections. This memo defines a simple protocol by which management information for a network element may be inspected or altered by logically remote users. In particular, together with its companion memos which describe the structure of management information along with the management information base, these documents provide a simple, workable architecture and system for managing TCP/IP-based internets and in particular the Internet.

The Internet Activities Board recommends that all IP and TCP implementations be network manageable. This implies implementation of the Internet MIB (RFC-1156) and at least one of the two recommended management protocols SNMP (RFC-1157) or CMOT (RFC-1095). It should be noted that, at this time, SNMP is a full Internet standard and CMOT is a draft standard. See also the Host and Gateway Requirements RFCs for more specific information on the applicability of this standard.

Please refer to the latest edition of the "IAB Official Protocol Standards" RFC for current information on the state and status of standard Internet protocols.

Distribution of this memo is unlimited.

2. Introduction

As reported in RFC 1052, IAB Recommendations for the Development of Internet Network Management Standards [1], a two-prong strategy for network management of TCP/IP-based internets was undertaken. In the short-term, the Simple Network Management Protocol (SNMP) was to be used to manage nodes in the Internet community. In the long-term, the use of the OSI network management framework was to be examined. Two documents were produced to define the management information: RFC 1065, which defined the Structure of Management Information (SMI) [2], and RFC 1066, which defined the Management Information Base (MIB) [3]. Both of these documents were designed so as to be compatible with both the SNMP and the OSI network management framework.

This strategy was quite successful in the short-term: Internet-based network management technology was fielded, by both the research and commercial communities, within a few months. As a result of this, portions of the Internet community became network manageable in a timely fashion.

As reported in RFC 1109, Report of the Second Ad Hoc Network Management Review Group [4], the requirements of the SNMP and the OSI network management frameworks were more different than anticipated. As such, the requirement for compatibility between the SMI/MIB and both frameworks was suspended. This action permitted the operational network management framework, the SNMP, to respond to new operational needs in the Internet community by producing documents defining new MIB items.

The IAB has designated the SNMP, SMI, and the initial Internet MIB to be full "Standard Protocols" with "Recommended" status. By this action, the IAB recommends that all IP and TCP implementations be network manageable and that the implementations that are network manageable are expected to adopt and implement the SMI, MIB, and SNMP.

As such, the current network management framework for TCP/IP- based internets consists of: Structure and Identification of Management Information for TCP/IP-based Internets, which describes how managed objects contained in the MIB are defined as set forth in RFC 1155 [5]; Management Information Base for Network Management of TCP/IP-based Internets, which describes the managed objects contained in the MIB as set forth in RFC 1156 [6]; and, the Simple Network Management Protocol, which defines the protocol used to manage these objects, as set forth in this memo.

As reported in RFC 1052, IAB Recommendations for the Development of Internet Network Management Standards [1], the Internet Activities Board has directed the Internet Engineering Task Force (IETF) to create two new working groups in the area of network management. One group was charged with the further specification and definition of elements to be included in the Management Information Base (MIB). The other was charged with defining the modifications to the Simple Network Management Protocol (SNMP) to accommodate the short-term needs of the network vendor and operations communities, and to align with the output of the MIB working group.

The MIB working group produced two memos, one which defines a Structure for Management Information (SMI) [2] for use by the managed objects contained in the MIB. A second memo [3] defines the list of managed objects.

The output of the SNMP Extensions working group is this memo, which incorporates changes to the initial SNMP definition [7] required to attain alignment with the output of the MIB working group. The changes should be minimal in order to be consistent with the IAB's directive that the working groups be "extremely sensitive to the need to keep the SNMP simple." Although considerable care and debate has gone into the changes to the SNMP which are reflected in this memo, the resulting protocol is not backwardly-compatible with its predecessor, the Simple Gateway Monitoring Protocol (SGMP) [8]. Although the syntax of the protocol has been altered, the original philosophy, design decisions, and architecture remain intact. In order to avoid confusion, new UDP ports have been allocated for use by the protocol described in this memo.

3. The SNMP Architecture

Implicit in the SNMP architectural model is a collection of network management stations and network elements. Network management stations execute management applications which monitor and control network elements. Network elements are devices such as hosts, gateways, terminal servers, and the like, which have management agents responsible for performing the network management functions requested by the network management stations. The Simple Network Management Protocol (SNMP) is used to communicate management information between the network management stations and the agents in the network elements.

3.1. Goals of the Architecture

The SNMP explicitly minimizes the number and complexity of management functions realized by the management agent itself. This goal is attractive in at least four respects:

(1) The development cost for management agent software necessary to support the protocol is accordingly reduced.

(2) The degree of management function that is remotely supported is accordingly increased, thereby admitting fullest use of internet resources in the management task.

(3) The degree of management function that is remotely supported is accordingly increased, thereby imposing the fewest possible restrictions on the form and sophistication of management tools.

(4) Simplified sets of management functions are easily understood and used by developers of network management tools.

A second goal of the protocol is that the functional paradigm for monitoring and control be sufficiently extensible to accommodate additional, possibly unanticipated aspects of network operation and management.

A third goal is that the architecture be, as much as possible, independent of the architecture and mechanisms of particular hosts or particular gateways.

3.2. Elements of the Architecture

The SNMP architecture articulates a solution to the network management problem in terms of:

(1) the scope of the management information communicated by the protocol,

(2) the representation of the management information communicated by the protocol,

(3) operations on management information supported by the protocol,

(4) the form and meaning of exchanges among management entities,

(5) the definition of administrative relationships among management entities, and

(6) the form and meaning of references to management information.

3.2.1. Scope of Management Information

The scope of the management information communicated by operation of the SNMP is exactly that represented by instances of all non-aggregate object types either defined in Internet-standard MIB or defined elsewhere according to the conventions set forth in Internet-standard SMI [5].

Support for aggregate object types in the MIB is neither required for conformance with the SMI nor realized by the SNMP.

3.2.2. Representation of Management Information

Management information communicated by operation of the SNMP is represented according to the subset of the ASN.1 language [9] that is specified for the definition of non-aggregate types in the SMI.

The SGMP adopted the convention of using a well-defined subset of the ASN.1 language [9]. The SNMP continues and extends this tradition by utilizing a moderately more complex subset of ASN.1 for describing managed objects and for describing the protocol data units used for managing those objects. In addition, the desire to ease eventual transition to OSI-based network management protocols led to the definition in the ASN.1 language of an Internet-standard Structure of Management Information (SMI) [5] and Management Information Base (MIB) [6]. The use of the ASN.1 language, was, in part, encouraged by the successful use of ASN.1 in earlier efforts, in particular, the SGMP. The restrictions on the use of ASN.1 that are part of the SMI contribute to the simplicity espoused and validated by experience with the SGMP.

Also for the sake of simplicity, the SNMP uses only a subset of the basic encoding rules of ASN.1 [10]. Namely, all encodings use the definite-length form. Further, whenever permissible, non-constructor encodings are used rather than constructor encodings. This restriction applies to all aspects of ASN.1 encoding, both for the top-level protocol data units and the data objects they contain.

3.2.3. Operations Supported on Management Information

The SNMP models all management agent functions as alterations or inspections of variables. Thus, a protocol entity on a logically remote host (possibly the network element itself) interacts with the management agent resident on the network element in order to retrieve (get) or alter (set) variables. This strategy has at least two positive consequences:

(1) It has the effect of limiting the number of essential management functions realized by the management agent to two: one operation to assign a value to a specified configuration or other parameter and another to retrieve such a value.

(2) A second effect of this decision is to avoid introducing into the protocol definition support for imperative management commands: the number of such commands is in practice ever-increasing, and the semantics of such commands are in general arbitrarily complex.

The strategy implicit in the SNMP is that the monitoring of network state at any significant level of detail is accomplished primarily by polling for appropriate information on the part of the monitoring center(s). A limited number of unsolicited messages (traps) guide the timing and focus of the polling. Limiting the number of unsolicited messages is consistent with the goal of simplicity and minimizing the amount of traffic generated by the network management function.

The exclusion of imperative commands from the set of explicitly supported management functions is unlikely to preclude any desirable management agent operation. Currently, most commands are requests either to set the value of some parameter or to retrieve such a value, and the function of the few imperative commands currently supported is easily accommodated in an asynchronous mode by this management model. In this scheme, an imperative command might be realized as the setting of a parameter value that subsequently triggers the desired action. For example, rather than implementing a "reboot command," this action might be invoked by simply setting a parameter indicating the number of seconds until system reboot.

3.2.4. Form and Meaning of Protocol Exchanges

The communication of management information among management entities is realized in the SNMP through the exchange of protocol messages. The form and meaning of those messages is defined below in Section 4.

Consistent with the goal of minimizing complexity of the management agent, the exchange of SNMP messages requires only an unreliable datagram service, and every message is entirely and independently represented by a single transport datagram. While this document specifies the exchange of messages via the UDP protocol [11], the mechanisms of the SNMP are generally suitable for use with a wide variety of transport services.

3.2.5. Definition of Administrative Relationships

The SNMP architecture admits a variety of administrative relationships among entities that participate in the protocol. The entities residing at management stations and network elements which communicate with one another using the SNMP are termed SNMP application entities. The peer processes which implement the SNMP, and thus support the SNMP application entities, are termed protocol entities.

A pairing of an SNMP agent with some arbitrary set of SNMP application entities is called an SNMP community. Each SNMP community is named by a string of octets, that is called the community name for said community.

An SNMP message originated by an SNMP application entity that in fact belongs to the SNMP community named by the community component of said message is called an authentic SNMP message. The set of rules by which an SNMP message is identified as an authentic SNMP message for a particular SNMP community is called an authentication scheme. An implementation of a function that identifies authentic SNMP messages according to one or more authentication schemes is called an authentication service.

Clearly, effective management of administrative relationships among SNMP application entities requires authentication services that (by the use of encryption or other techniques) are able to identify authentic SNMP messages with a high degree of certainty. Some SNMP implementations may wish to support only a trivial authentication service that identifies all SNMP messages as authentic SNMP messages.

For any network element, a subset of objects in the MIB that pertain to that element is called a SNMP MIB view. Note that the names of the object types represented in a SNMP MIB view need not belong to a single sub-tree of the object type name space.

An element of the set { READ-ONLY, READ-WRITE } is called an SNMP access mode.

A pairing of a SNMP access mode with a SNMP MIB view is called an SNMP community profile. A SNMP community profile represents specified access privileges to variables in a specified MIB view. For every variable in the MIB view in a given SNMP community profile, access to that variable is represented by the profile according to the following conventions:

(1) if said variable is defined in the MIB with "Access:" of "none," it is unavailable as an operand for any operator;

(2) if said variable is defined in the MIB with "Access:" of "read-write" or "write-only" and the access mode of the given profile is READ-WRITE, that variable is available as an operand for the get, set, and trap operations;

(3) otherwise, the variable is available as an operand for the get and trap operations.

(4) In those cases where a "write-only" variable is an operand used for the get or trap operations, the value given for the variable is implementation-specific.

A pairing of a SNMP community with a SNMP community profile is called a SNMP access policy. An access policy represents a specified community profile afforded by the SNMP agent of a specified SNMP community to other members of that community. All administrative relationships among SNMP application entities are architecturally defined in terms of SNMP access policies.

For every SNMP access policy, if the network element on which the SNMP agent for the specified SNMP community resides is not that to which the MIB view for the specified profile pertains, then that policy is called a SNMP proxy access policy. The SNMP agent associated with a proxy access policy is called a SNMP proxy agent. While careless definition of proxy access policies can result in management loops, prudent definition of proxy policies is useful in at least two ways:

(1) It permits the monitoring and control of network elements which are otherwise not addressable using the management protocol and the transport protocol. That is, a proxy agent may provide a protocol conversion function allowing a management station to apply a consistent management

framework to all network elements, including devices such as modems, multiplexors, and other devices which support different management frameworks.

(2) It potentially shields network elements from elaborate access control policies. For example, a proxy agent may implement sophisticated access control whereby diverse subsets of variables within the MIB are made accessible to different management stations without increasing the complexity of the network element.

By way of example, Figure 1 illustrates the relationship between management stations, proxy agents, and management agents. In this example, the proxy agent is envisioned to be a normal Internet Network Operations Center (INOC) of some administrative domain which has a standard managerial relationship with a set of management agents.

```
   +------------------+         +----------------+        +----------------+
   |  Region #1 INOC  |         |Region #2 INOC  |        |PC in Region #3 |
   |                  |         |                |        |                |
   |Domain=Region #1  |         |Domain=Region #2|        |Domain=Region #3|
   |CPU=super-mini-1  |         |CPU=super-mini-1|        |CPU=Clone-1     |
   |PCommunity=pub    |         |PCommunity=pub  |        |PCommunity=slate|
   |                  |         |                |        |                |
   +------------------+         +----------------+        +----------------+
           /|\                        /|\                       /|\
            |                          |                         |
            |                          |                         |
            |                         \|/                        |
            |                +-----------------+                 |
            +--------------->| Region #3 INOC  |<----------------+
                             |                 |
                             |Domain=Region #3 |
                             |CPU=super-mini-2 |
                             |PCommunity=pub,  |
                             |         slate   |
                             |DCommunity=secret|
            +--------------->|                 |<----------------+
            |                +-----------------+                 |
            |                         /|\                        |
            |                          |                         |
            |                          |                         |
           \|/                        \|/                       \|/
   +-----------------+       +-----------------+        +-----------------+
   |Domain=Region#3  |       |Domain=Region#3  |        |Domain=Region#3  |
   |CPU=router-1     |       |CPU=mainframe-1  |        |CPU=modem-1      |
   |DCommunity=secret|       |DCommunity=secret|        |DCommunity=secret|
   +-----------------+       +-----------------+        +-----------------+
```

Domain: the administrative domain of the element
PCommunity: the name of a community utilizing a proxy agent
DCommunity: the name of a direct community

Figure 1
Example Network Management Configuration

3.2.6. Form and Meaning of References to Managed Objects

The SMI requires that the definition of a conformant management protocol address:

(1) the resolution of ambiguous MIB references,

(2) the resolution of MIB references in the presence multiple MIB versions, and

(3) the identification of particular instances of object types defined in the MIB.

3.2.6.1. Resolution of Ambiguous MIB References

Because the scope of any SNMP operation is conceptually confined to objects relevant to a single network element, and because all SNMP references to MIB objects are (implicitly or explicitly) by unique variable names, there is no possibility that any SNMP reference to any object type defined in the MIB could resolve to multiple instances of that type.

3.2.6.2. Resolution of References across MIB Versions

The object instance referred to by any SNMP operation is exactly that specified as part of the operation request or (in the case of a get-next operation) its immediate successor in the MIB as a whole. In particular, a reference to an object as part of some version of the Internet-standard MIB does not resolve to any object that is not part of said version of the Internet-standard MIB, except in the case that the requested operation is get-next and the specified object name is lexicographically last among the names of all objects presented as part of said version of the Internet-Standard MIB.

3.2.6.3. Identification of Object Instances

The names for all object types in the MIB are defined explicitly either in the Internet-standard MIB or in other documents which conform to the naming conventions of the SMI. The SMI requires that conformant management protocols define mechanisms for identifying individual instances of those object types for a particular network element.

Each instance of any object type defined in the MIB is identified in SNMP operations by a unique name called its "variable name." In general, the name of an SNMP variable is an OBJECT IDENTIFIER of the form x.y, where x is the name of a non-aggregate object type defined in the MIB and y is an OBJECT IDENTIFIER fragment that, in a way specific to the named object type, identifies the desired instance.

This naming strategy admits the fullest exploitation of the semantics of the GetNextRequest-PDU (see Section 4), because it assigns names for related variables so as to be contiguous in the lexicographical ordering of all variable names known in the MIB.

The type-specific naming of object instances is defined below for a number of classes of object types. Instances of an object type to which none of the following naming conventions are applicable are named by OBJECT IDENTIFIERs of the form x.0, where x is the name of said object type in the MIB definition.

For example, suppose one wanted to identify an instance of the variable sysDescr The object class for sysDescr is:

```
iso org dod internet mgmt mib system sysDescr
 1   3   6     1       2    1    1       1
```

Hence, the object type, x, would be 1.3.6.1.2.1.1.1 to which is appended an instance sub-identifier of 0. That is, 1.3.6.1.2.1.1.1.0 identifies the one and only instance of sysDescr.

3.2.6.3.1. ifTable Object Type Names

The name of a subnet interface, s, is the OBJECT IDENTIFIER value of the form i, where i has the value of that instance of the ifIndex object type associated with s.

For each object type, t, for which the defined name, n, has a prefix of ifEntry, an instance, i, of t is named by an OBJECT IDENTIFIER of the form n.s, where s is the name of the subnet interface about which i represents information.

For example, suppose one wanted to identify the instance of the variable ifType associated with interface 2. Accordingly, ifType.2 would identify the desired instance.

3.2.6.3.2. atTable Object Type Names

The name of an AT-cached network address, x, is an OBJECT IDENTIFIER of the form 1.a.b.c.d, where a.b.c.d is the value (in the familiar "dot" notation) of the atNetAddress object type associated with x.

The name of an address translation equivalence e is an OBJECT IDENTIFIER value of the form s.w, such that s is the value of that instance of the atIndex object type associated with e and such that w is the name of the AT-cached network address associated with e.

For each object type, t, for which the defined name, n, has a prefix of atEntry, an instance, i, of t is named by an OBJECT IDENTIFIER of the form n.y, where y is the name of the address translation equivalence about which i represents information.

For example, suppose one wanted to find the physical address of an entry in the address translation table (ARP cache) associated with an IP address of 89.1.1.42 and interface 3. Accordingly, atPhysAddress.3.1.89.1.1.42 would identify the desired instance.

3.2.6.3.3. ipAddrTable Object Type Names

The name of an IP-addressable network element, x, is the OBJECT IDENTIFIER of the form a.b.c.d such that a.b.c.d is the value (in the familiar "dot" notation) of that instance of the ipAdEntAddr object type associated with x.

For each object type, t, for which the defined name, n, has a prefix of ipAddrEntry, an instance, i, of t is named by an OBJECT IDENTIFIER of the form n.y, where y is the name of the IP-addressable network element about which i represents information.

For example, suppose one wanted to find the network mask of an entry in the IP interface table associated with an IP address of 89.1.1.42. Accordingly, ipAdEntNetMask.89.1.1.42 would identify the desired instance.

3.2.6.3.4. ipRoutingTable Object Type Names

The name of an IP route, x, is the OBJECT IDENTIFIER of the form a.b.c.d such that a.b.c.d is the value (in the familiar "dot" notation) of that instance of the ipRouteDest object type associated with x.

For each object type, t, for which the defined name, n, has a prefix of ipRoutingEntry, an instance, i, of t is named by an OBJECT IDENTIFIER of the form n.y, where y is the name of the IP route about which i represents information.

For example, suppose one wanted to find the next hop of an entry in the IP routing table associated with the destination of 89.1.1.42. Accordingly, ipRouteNextHop.89.1.1.42 would identify the desired instance.

3.2.6.3.5. tcpConnTable Object Type Names

The name of a TCP connection, x, is the OBJECT IDENTIFIER of the form a.b.c.d.e.f.g.h.i.j such that a.b.c.d is the value (in the familiar "dot" notation) of that instance of the tcpConnLocalAddress object type associated with x and such that f.g.h.i is the value (in the familiar "dot" notation) of that instance of the tcpConnRemoteAddress object type associated with x and such that e is the value of that instance of the tcpConnLocalPort object type associated with x and such that j is the value of that instance of the tcpConnRemotePort object type associated with x.

For each object type, t, for which the defined name, n, has a prefix of tcpConnEntry, an instance, i, of t is named by an OBJECT IDENTIFIER of the form n.y, where y is the name of the TCP connection about which i represents information.

For example, suppose one wanted to find the state of a TCP connection between the local address of 89.1.1.42 on TCP port 21 and the remote address of 10.0.0.51 on TCP port 2059. Accordingly, tcpConnState.89.1.1.42.21.10.0.0.51.2059 would identify the desired instance.

3.2.6.3.6. egpNeighTable Object Type Names

The name of an EGP neighbor, x, is the OBJECT IDENTIFIER of the form a.b.c.d such that a.b.c.d is the value (in the familiar "dot" notation) of that instance of the egpNeighAddr object type associated with x.

For each object type, t, for which the defined name, n, has a prefix of egpNeighEntry, an instance, i, of t is named by an OBJECT IDENTIFIER of the form n.y, where y is the name of the EGP neighbor about which i represents information.

For example, suppose one wanted to find the neighbor state for the IP address of 89.1.1.42. Accordingly, egpNeighState.89.1.1.42 would identify the desired instance.

4. Protocol Specification

The network management protocol is an application protocol by which the variables of an agent's MIB may be inspected or altered.

Communication among protocol entities is accomplished by the exchange of messages, each of which is entirely and independently represented within a single UDP datagram using the basic encoding rules of ASN.1 (as discussed in Section 3.2.2). A message consists of a version identifier, an SNMP community name, and a protocol data unit (PDU). A protocol entity receives messages at UDP port 161 on the host with which it is associated for all messages except for those which report traps (i.e., all messages except those which contain the Trap-PDU). Messages which report traps should be received on UDP port 162 for further processing. An implementation of this protocol need not accept messages whose length exceeds 484 octets. However, it is recommended that implementations support larger datagrams whenever feasible.

It is mandatory that all implementations of the SNMP support the five PDUs: GetRequest-PDU, GetNextRequest-PDU, GetResponse-PDU, SetRequest-PDU, and Trap-PDU.

```
 RFC1157-SNMP DEFINITIONS ::= BEGIN

  IMPORTS
       ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
               FROM RFC1155-SMI;


  -- top-level message

          Message ::=
                  SEQUENCE {
                       version        -- version-1 for this RFC
                          INTEGER {
                              version-1(0)
                          },

                      community      -- community name
                          OCTET STRING,

                      data           -- e.g., PDUs if trivial
                          ANY        -- authentication is being used
                  }
```

```
-- protocol data units

        PDUs ::=
                CHOICE {
                    get-request
                        GetRequest-PDU,

                    get-next-request
                        GetNextRequest-PDU,

                    get-response
                        GetResponse-PDU,

                    set-request
                        SetRequest-PDU,

                    trap
                        Trap-PDU
                     }

-- the individual PDUs and commonly used
-- data types will be defined later

END
```

4.1. Elements of Procedure

This section describes the actions of a protocol entity implementing the SNMP. Note, however, that it is not intended to constrain the internal architecture of any conformant implementation.

In the text that follows, the term transport address is used. In the case of the UDP, a transport address consists of an IP address along with a UDP port. Other transport services may be used to support the SNMP. In these cases, the definition of a transport address should be made accordingly.

The top-level actions of a protocol entity which generates a message are as follows:

(1) It first constructs the appropriate PDU, e.g., the GetRequest-PDU, as an ASN.1 object.

(2) It then passes this ASN.1 object along with a community name its source transport address and the destination transport address, to the service which implements the desired authentication scheme. This authentication

service returns another ASN.1 object.

(3) The protocol entity then constructs an ASN.1 Message object, using the community name and the resulting ASN.1 object.

(4) This new ASN.1 object is then serialized, using the basic encoding rules of ASN.1, and then sent using a transport service to the peer protocol entity.

Similarly, the top-level actions of a protocol entity which receives a message are as follows:

(1) It performs a rudimentary parse of the incoming datagram to build an ASN.1 object corresponding to an ASN.1 Message object. If the parse fails, it discards the datagram and performs no further actions.

(2) It then verifies the version number of the SNMP message. If there is a mismatch, it discards the datagram and performs no further actions.

(3) The protocol entity then passes the community name and user data found in the ASN.1 Message object, along with the datagram's source and destination transport addresses to the service which implements the desired authentication scheme. This entity returns another ASN.1 object, or signals an authentication failure. In the latter case, the protocol entity notes this failure, (possibly) generates a trap, and discards the datagram and performs no further actions.

(4) The protocol entity then performs a rudimentary parse on the ASN.1 object returned from the authentication service to build an ASN.1 object corresponding to an ASN.1 PDUs object. If the parse fails, it discards the datagram and performs no further actions. Otherwise, using the named SNMP community, the appropriate profile is selected, and the PDU is processed accordingly. If, as a result of this processing, a message is returned then the source transport address that the response message is sent from shall be identical to the destination transport address that the original request message was sent to.

4.1.1. Common Constructs

Before introducing the six PDU types of the protocol, it is appropriate to consider some of the ASN.1 constructs used frequently:

```
-- request/response information

RequestID ::=
        INTEGER

ErrorStatus ::=
        INTEGER {
            noError(0),
            tooBig(1),
            noSuchName(2),
            badValue(3),
            readOnly(4)
            genErr(5)
        }

ErrorIndex ::=
        INTEGER


-- variable bindings

VarBind ::=
        SEQUENCE {
            name
                ObjectName,

            value
                ObjectSyntax
        }

VarBindList ::=
        SEQUENCE OF
            VarBind
```

RequestIDs are used to distinguish among outstanding requests. By use of the RequestID, an SNMP application entity can correlate incoming responses with outstanding requests. In cases where an unreliable datagram service is being used, the RequestID also provides a simple means of identifying messages duplicated by the network.

A non-zero instance of ErrorStatus is used to indicate that an exception occurred while processing a request. In these cases, ErrorIndex may provide additional information by indicating which variable in a list caused the exception.

The term variable refers to an instance of a managed object. A variable binding, or VarBind, refers to the pairing of the name of a variable to the variable's value. A VarBindList is a simple list of variable names and corresponding values. Some PDUs are concerned only with the name of a variable and not its value (e.g., the GetRequest-PDU). In this case, the value portion of the binding is ignored by the protocol entity. However, the value portion must still have valid ASN.1 syntax and encoding. It is recommended that the ASN.1 value NULL be used for the value portion of such bindings.

4.1.2. The GetRequest-PDU

The form of the GetRequest-PDU is:

```
GetRequest-PDU ::=
    [0]
        IMPLICIT SEQUENCE {
            request-id
                RequestID,

            error-status        -- always 0
                ErrorStatus,

            error-index         -- always 0
                ErrorIndex,

            variable-bindings
                VarBindList
        }
```

The GetRequest-PDU is generated by a protocol entity only at the request of its SNMP application entity.

Upon receipt of the GetRequest-PDU, the receiving protocol entity responds according to any applicable rule in the list below:

(1) If, for any object named in the variable-bindings field, the object's name does not exactly match the name of some object available for get operations in the relevant MIB view, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is noSuchName, and the value of the error-index field is the index of said object name component in the received

message.

(2) If, for any object named in the variable-bindings field, the object is an aggregate type (as defined in the SMI), then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is noSuchName, and the value of the error-index field is the index of said object name component in the received message.

(3) If the size of the GetResponse-PDU generated as described below would exceed a local limitation, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is tooBig, and the value of the error-index field is zero.

(4) If, for any object named in the variable-bindings field, the value of the object cannot be retrieved for reasons not covered by any of the foregoing rules, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is genErr and the value of the error-index field is the index of said object name component in the received message.

If none of the foregoing rules apply, then the receiving protocol entity sends to the originator of the received message the GetResponse-PDU such that, for each object named in the variable-bindings field of the received message, the corresponding component of the GetResponse-PDU represents the name and value of that variable. The value of the error- status field of the GetResponse-PDU is noError and the value of the error-index field is zero. The value of the request-id field of the GetResponse-PDU is that of the received message.

4.1.3. The GetNextRequest-PDU

The form of the GetNextRequest-PDU is identical to that of the GetRequest-PDU except for the indication of the PDU type. In the ASN.1 language:

```
GetNextRequest-PDU ::=
    [1]
        IMPLICIT SEQUENCE {
            request-id
                RequestID,
```

```
                      error-status          -- always 0
                          ErrorStatus,

                      error-index           -- always 0
                          ErrorIndex,

                      variable-bindings
                          VarBindList
                  }
```

The GetNextRequest-PDU is generated by a protocol entity only at the request of its SNMP application entity.

Upon receipt of the GetNextRequest-PDU, the receiving protocol entity responds according to any applicable rule in the list below:

(1) If, for any object name in the variable-bindings field, that name does not lexicographically precede the name of some object available for get operations in the relevant MIB view, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is noSuchName, and the value of the error-index field is the index of said object name component in the received message.

(2) If the size of the GetResponse-PDU generated as described below would exceed a local limitation, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is tooBig, and the value of the error-index field is zero.

(3) If, for any object named in the variable-bindings field, the value of the lexicographical successor to the named object cannot be retrieved for reasons not covered by any of the foregoing rules, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is genErr and the value of the error-index field is the index of said object name component in the received message.

If none of the foregoing rules apply, then the receiving protocol entity sends to the originator of the received message the GetResponse-PDU such that, for each name in the variable-bindings field of the received message, the corresponding component of the

GetResponse-PDU represents the name and value of that object whose name is, in the lexicographical ordering of the names of all objects available for get operations in the relevant MIB view, together with the value of the name field of the given component, the immediate successor to that value. The value of the error-status field of the GetResponse-PDU is noError and the value of the errorindex field is zero. The value of the request-id field of the GetResponse-PDU is that of the received message.

4.1.3.1. Example of Table Traversal

One important use of the GetNextRequest-PDU is the traversal of conceptual tables of information within the MIB. The semantics of this type of SNMP message, together with the protocol-specific mechanisms for identifying individual instances of object types in the MIB, affords access to related objects in the MIB as if they enjoyed a tabular organization.

By the SNMP exchange sketched below, an SNMP application entity might extract the destination address and next hop gateway for each entry in the routing table of a particular network element. Suppose that this routing table has three entries:

Destination	NextHop	Metric
10.0.0.99	89.1.1.42	5
9.1.2.3	99.0.0.3	3
10.0.0.51	89.1.1.42	5

The management station sends to the SNMP agent a GetNextRequest-PDU containing the indicated OBJECT IDENTIFIER values as the requested variable names:

```
GetNextRequest ( ipRouteDest, ipRouteNextHop, ipRouteMetric1 )
```

The SNMP agent responds with a GetResponse-PDU:

```
GetResponse (( ipRouteDest.9.1.2.3 =  "9.1.2.3" ),
        ( ipRouteNextHop.9.1.2.3 = "99.0.0.3" ),
        ( ipRouteMetric1.9.1.2.3 = 3 ))
```

The management station continues with:

```
GetNextRequest ( ipRouteDest.9.1.2.3,
        ipRouteNextHop.9.1.2.3,
```

```
                    ipRouteMetric1.9.1.2.3 )
```

The SNMP agent responds:

```
              GetResponse (( ipRouteDest.10.0.0.51 = "10.0.0.51" ),
                      ( ipRouteNextHop.10.0.0.51 = "89.1.1.42" ),
                      ( ipRouteMetric1.10.0.0.51 = 5 ))
```

The management station continues with:

```
              GetNextRequest ( ipRouteDest.10.0.0.51,
                      ipRouteNextHop.10.0.0.51,
                      ipRouteMetric1.10.0.0.51 )
```

The SNMP agent responds:

```
              GetResponse (( ipRouteDest.10.0.0.99 = "10.0.0.99" ),
                      ( ipRouteNextHop.10.0.0.99 = "89.1.1.42" ),
                      ( ipRouteMetric1.10.0.0.99 = 5 ))
```

The management station continues with:

```
              GetNextRequest ( ipRouteDest.10.0.0.99,
                      ipRouteNextHop.10.0.0.99,
                      ipRouteMetric1.10.0.0.99 )
```

As there are no further entries in the table, the SNMP agent returns those objects that are next in the lexicographical ordering of the known object names. This response signals the end of the routing table to the management station.

4.1.4. The GetResponse-PDU

The form of the GetResponse-PDU is identical to that of the GetRequest-PDU except for the indication of the PDU type. In the ASN.1 language:

```
          GetResponse-PDU ::=
              [2]
                  IMPLICIT SEQUENCE {
                      request-id
                          RequestID,
```

```
                        error-status
                            ErrorStatus,

                        error-index
                            ErrorIndex,

                        variable-bindings
                            VarBindList
                    }
```

The GetResponse-PDU is generated by a protocol entity only upon receipt of the GetRequest-PDU, GetNextRequest-PDU, or SetRequest-PDU, as described elsewhere in this document.

Upon receipt of the GetResponse-PDU, the receiving protocol entity presents its contents to its SNMP application entity.

4.1.5. The SetRequest-PDU

The form of the SetRequest-PDU is identical to that of the GetRequest-PDU except for the indication of the PDU type. In the ASN.1 language:

```
          SetRequest-PDU ::=
              [3]
                  IMPLICIT SEQUENCE {
                      request-id
                          RequestID,

                      error-status        -- always 0
                          ErrorStatus,

                      error-index         -- always 0
                          ErrorIndex,

                      variable-bindings
                          VarBindList
                  }
```

The SetRequest-PDU is generated by a protocol entity only at the request of its SNMP application entity.

Upon receipt of the SetRequest-PDU, the receiving entity responds according to any applicable rule in the list below:

(1) If, for any object named in the variable-bindings field,

the object is not available for set operations in the relevant MIB view, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is noSuchName, and the value of the error-index field is the index of said object name component in the received message.

(2) If, for any object named in the variable-bindings field, the contents of the value field does not, according to the ASN.1 language, manifest a type, length, and value that is consistent with that required for the variable, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is badValue, and the value of the error-index field is the index of said object name in the received message.

(3) If the size of the Get Response type message generated as described below would exceed a local limitation, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is tooBig, and the value of the error-index field is zero.

(4) If, for any object named in the variable-bindings field, the value of the named object cannot be altered for reasons not covered by any of the foregoing rules, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is genErr and the value of the error-index field is the index of said object name component in the received message.

If none of the foregoing rules apply, then for each object named in the variable-bindings field of the received message, the corresponding value is assigned to the variable. Each variable assignment specified by the SetRequest-PDU should be effected as if simultaneously set with respect to all other assignments specified in the same message.

The receiving entity then sends to the originator of the received message the GetResponse-PDU of identical form except that the value of the error-status field of the generated message is noError and the value of the error-index field is zero.

4.1.6. The Trap-PDU

The form of the Trap-PDU is:

```
  Trap-PDU ::=
      [4]

           IMPLICIT SEQUENCE {
              enterprise          -- type of object generating
                                  -- trap, see sysObjectID in [5]
                  OBJECT IDENTIFIER,

              agent-addr          -- address of object generating
                  NetworkAddress, -- trap

              generic-trap        -- generic trap type
                  INTEGER {
                      coldStart(0),
                      warmStart(1),
                      linkDown(2),
                      linkUp(3),
                      authenticationFailure(4),
                      egpNeighborLoss(5),
                      enterpriseSpecific(6)
                  },

              specific-trap     -- specific code, present even
                  INTEGER,      -- if generic-trap is not
                                -- enterpriseSpecific

              time-stamp        -- time elapsed between the last
                TimeTicks,      -- (re)initialization of the network
                                -- entity and the generation of the
                                   trap

              variable-bindings   -- "interesting" information
                   VarBindList
          }
```

The Trap-PDU is generated by a protocol entity only at the request of the SNMP application entity. The means by which an SNMP application entity selects the destination addresses of the SNMP application entities is implementation-specific.

Upon receipt of the Trap-PDU, the receiving protocol entity presents its contents to its SNMP application entity.

The significance of the variable-bindings component of the Trap-PDU is implementation-specific.

Interpretations of the value of the generic-trap field are:

4.1.6.1. The coldStart Trap

A coldStart(0) trap signifies that the sending protocol entity is reinitializing itself such that the agent's configuration or the protocol entity implementation may be altered.

4.1.6.2. The warmStart Trap

A warmStart(1) trap signifies that the sending protocol entity is reinitializing itself such that neither the agent configuration nor the protocol entity implementation is altered.

4.1.6.3. The linkDown Trap

A linkDown(2) trap signifies that the sending protocol entity recognizes a failure in one of the communication links represented in the agent's configuration.

The Trap-PDU of type linkDown contains as the first element of its variable-bindings, the name and value of the ifIndex instance for the affected interface.

4.1.6.4. The linkUp Trap

A linkUp(3) trap signifies that the sending protocol entity recognizes that one of the communication links represented in the agent's configuration has come up.

The Trap-PDU of type linkUp contains as the first element of its variable-bindings, the name and value of the ifIndex instance for the affected interface.

4.1.6.5. The authenticationFailure Trap

An authenticationFailure(4) trap signifies that the sending protocol entity is the addressee of a protocol message that is not properly authenticated. While implementations of the SNMP must be capable of generating this trap, they must also be capable of suppressing the emission of such traps via an implementation-specific mechanism.

4.1.6.6. The egpNeighborLoss Trap

An egpNeighborLoss(5) trap signifies that an EGP neighbor for whom the sending protocol entity was an EGP peer has been marked down and the peer relationship no longer obtains.

The Trap-PDU of type egpNeighborLoss contains as the first element of its variable-bindings, the name and value of the egpNeighAddr instance for the affected neighbor.

4.1.6.7. The enterpriseSpecific Trap

A enterpriseSpecific(6) trap signifies that the sending protocol entity recognizes that some enterprise-specific event has occurred. The specific-trap field identifies the particular trap which occurred.

5. Definitions

```
RFC1157-SNMP DEFINITIONS ::= BEGIN

 IMPORTS
     ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
         FROM RFC1155-SMI;


     -- top-level message

     Message ::=
             SEQUENCE {
                 version          -- version-1 for this RFC
                     INTEGER {
                         version-1(0)
                     },

                 community        -- community name
                     OCTET STRING,

                 data             -- e.g., PDUs if trivial
                     ANY          -- authentication is being used
             }


     -- protocol data units

     PDUs ::=
             CHOICE {
                         get-request
                             GetRequest-PDU,

                         get-next-request
                             GetNextRequest-PDU,

                         get-response
                             GetResponse-PDU,

                         set-request
                             SetRequest-PDU,

                         trap
                             Trap-PDU
                     }
```

```
-- PDUs

GetRequest-PDU ::=
    [0]
        IMPLICIT PDU

GetNextRequest-PDU ::=
    [1]
        IMPLICIT PDU

GetResponse-PDU ::=
    [2]
        IMPLICIT PDU

SetRequest-PDU ::=
    [3]
        IMPLICIT PDU

PDU ::=
        SEQUENCE {
           request-id
                INTEGER,

            error-status      -- sometimes ignored
                INTEGER {
                    noError(0),
                    tooBig(1),
                    noSuchName(2),
                    badValue(3),
                    readOnly(4),
                    genErr(5)
                },

            error-index       -- sometimes ignored
               INTEGER,

            variable-bindings -- values are sometimes ignored
                VarBindList
        }

Trap-PDU ::=
    [4]
       IMPLICIT SEQUENCE {
            enterprise        -- type of object generating
                              -- trap, see sysObjectID in [5]


                OBJECT IDENTIFIER,
```

```
                agent-addr        -- address of object generating
                    NetworkAddress, -- trap

                generic-trap      -- generic trap type
                    INTEGER {
                        coldStart(0),
                        warmStart(1),
                        linkDown(2),
                        linkUp(3),
                        authenticationFailure(4),
                        egpNeighborLoss(5),
                        enterpriseSpecific(6)
                    },

                specific-trap  -- specific code, present even
                    INTEGER,   -- if generic-trap is not
                               -- enterpriseSpecific

                time-stamp     -- time elapsed between the last
                    TimeTicks, -- (re)initialization of the
                                  network
                               -- entity and the generation of the
                                  trap

                 variable-bindings -- "interesting" information
                    VarBindList
            }

     -- variable bindings

     VarBind ::=
             SEQUENCE {
                 name
                     ObjectName,

                 value
                     ObjectSyntax
             }

    VarBindList ::=
            SEQUENCE OF
               VarBind

    END
```

7. References

[1] Cerf, V., "IAB Recommendations for the Development of Internet Network Management Standards", RFC 1052, IAB, April 1988.

[2] Rose, M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based internets", RFC 1065, TWG, August 1988.

[3] McCloghrie, K., and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets", RFC 1066, TWG, August 1988.

[4] Cerf, V., "Report of the Second Ad Hoc Network Management Review Group", RFC 1109, IAB, August 1989.

[5] Rose, M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based Internets", RFC 1155, Performance Systems International and Hughes LAN Systems, May 1990.

[6] McCloghrie, K., and M. Rose, "Management Information Base for Network Management of TCP/IP-based Internets", RFC 1156, Hughes LAN Systems and Performance Systems International, May 1990.

[7] Case, J., M. Fedor, M. Schoffstall, and J. Davin, "A Simple Network Management Protocol", Internet Engineering Task Force working note, Network Information Center, SRI International, Menlo Park, California, March 1988.

[8] Davin, J., J. Case, M. Fedor, and M. Schoffstall, "A Simple Gateway Monitoring Protocol", RFC 1028, Proteon, University of Tennessee at Knoxville, Cornell University, and Rensselaer Polytechnic Institute, November 1987.

[9] Information processing systems - Open Systems Interconnection, "Specification of Abstract Syntax Notation One (ASN.1)", International Organization for Standardization, International Standard 8824, December 1987.

[10] Information processing systems - Open Systems Interconnection, "Specification of Basic Encoding Rules for Abstract Notation One (ASN.1)", International

Organization for Standardization, International Standard 8825, December 1987.

[11] Postel, J., "User Datagram Protocol", RFC 768, USC/Information Sciences Institute, November 1980.

Security Considerations

Security issues are not discussed in this memo.

Authors' Addresses

Jeffrey D. Case
SNMP Research
P.O. Box 8593
Knoxville, TN 37996-4800

Phone: (615) 573-1434

Email: case@CS.UTK.EDU

Mark Fedor
Performance Systems International
Rensselaer Technology Park
125 Jordan Road
Troy, NY 12180

Phone: (518) 283-8860

Email: fedor@patton.NYSER.NET

Martin Lee Schoffstall
Performance Systems International
Rensselaer Technology Park
165 Jordan Road
Troy, NY 12180

Phone: (518) 283-8860

Email: schoff@NISC.NYSER.NET

James R. Davin
MIT Laboratory for Computer Science, NE43-507
545 Technology Square
Cambridge, MA 02139

Phone: (617) 253-6020

EMail: jrd@ptt.lcs.mit.edu

6. Acknowledgements

This memo was influenced by the IETF SNMP Extensions working group:

Karl Auerbach, Epilogue Technology
K. Ramesh Babu, Excelan
Amatzia Ben-Artzi, 3Com/Bridge
Lawrence Besaw, Hewlett-Packard
Jeffrey D. Case, University of Tennessee at Knoxville
Anthony Chung, Sytek
James Davidson, The Wollongong Group
James R. Davin, MIT Laboratory for Computer Science
Mark S. Fedor, NYSERNet
Phill Gross, The MITRE Corporation
Satish Joshi, ACC
Dan Lynch, Advanced Computing Environments
Keith McCloghrie, The Wollongong Group
Marshall T. Rose, The Wollongong Group (chair)
Greg Satz, cisco
Martin Lee Schoffstall, Rensselaer Polytechnic Institute
Wengyik Yeong, NYSERNet

Network Working Group
Request for Comments: 1213
Obsoletes: RFC 1158

K. McCloghrie
Hughes LAN Systems, Inc.
M. Rose
Performance Systems International
Editors

March 1991

Management Information Base for Network Management of TCP/IP-based internets: MIB-II

Status of this Memo

This memo defines the second version of the Management Information Base (MIB-II) for use with network management protocols in TCP/IP-based internets. This RFC specifies an IAB standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "IAB Official Protocol Standards" for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Table of Contents

1. Abstract

This memo defines the second version of the Management Information Base (MIB-II) for use with network management protocols in TCP/IP-based internets. In particular, together with its companion memos which describe the structure of management information (RFC 1155) along with the network management protocol (RFC 1157) for TCP/IP-based internets, these documents provide a simple, workable architecture and system for managing TCP/IP-based internets and in particular the Internet community.

2. Introduction

As reported in RFC 1052, IAB Recommendations for the Development of Internet Network Management Standards [1], a two-prong strategy for network management of TCP/IP-based internets was undertaken. In the short-term, the Simple Network Management Protocol (SNMP) was to be used to manage nodes in the Internet community. In the long-term, the use of the OSI network management framework was to be examined. Two documents were produced to define the management information: RFC 1065, which defined the Structure of Management Information (SMI) [2], and RFC 1066, which defined the Management Information Base (MIB) [3]. Both of these documents were designed so as to be compatible with both the SNMP and the OSI network management framework.

This strategy was quite successful in the short-term: Internet-based network management technology was fielded, by both the research and commercial communities, within a few months. As a result of this, portions of the Internet community became network manageable in a timely fashion.

As reported in RFC 1109, Report of the Second Ad Hoc Network Management Review Group [4], the requirements of the SNMP and the OSI network management frameworks were more different than anticipated. As such, the requirement for compatibility between the SMI/MIB and both frameworks was suspended. This action permitted the operational network management framework, the SNMP, to respond to new operational needs in the Internet community by producing this document.

As such, the current network management framework for TCP/IP- based internets consists of: Structure and Identification of Management Information for TCP/IP-based internets, RFC 1155 [12], which describes how managed objects contained in the MIB are defined; Management Information Base for Network Management of TCP/IP-based internets: MIB-II, this memo, which describes the managed objects contained in the MIB (and supercedes RFC 1156 [13]); and, the Simple Network Management Protocol, RFC 1098 [5], which defines the protocol used to manage these objects.

3. Changes from RFC 1156

Features of this MIB include:

(1) incremental additions to reflect new operational requirements;

(2) upwards compatibility with the SMI/MIB and the SNMP;

(3) improved support for multi-protocol entities; and,

(4) textual clean-up of the MIB to improve clarity and readability.

The objects defined in MIB-II have the OBJECT IDENTIFIER prefix:

```
mib-2      OBJECT IDENTIFIER ::= { mgmt 1 }
```

which is identical to the prefix used in MIB-I.

3.1. Deprecated Objects

In order to better prepare implementors for future changes in the MIB, a new term "deprecated" may be used when describing an object. A deprecated object in the MIB is one which must be supported, but one which will most likely be removed from the next version of the MIB (e.g., MIB-III).

MIB-II marks one object as being deprecated:

```
atTable
```

As a result of deprecating the atTable object, the entire Address Translation group is deprecated.

Note that no functionality is lost with the deprecation of these objects: new objects providing equivalent or superior functionality are defined in MIB-II.

3.2. Display Strings

In the past, there have been misinterpretations of the MIB as to when a string of octets should contain printable characters, meant to be displayed to a human. As a textual convention in the MIB, the datatype

```
DisplayString ::=
    OCTET STRING
```

is introduced. A DisplayString is restricted to the NVT ASCII character set, as defined in pages 10-11 of [6].

The following objects are now defined in terms of DisplayString:

```
sysDescr
ifDescr
```

It should be noted that this change has no effect on either the syntax nor semantics of these objects. The use of the DisplayString notation is merely an artifact of the explanatory method used in MIB-II and future MIBs.

Further it should be noted that any object defined in terms of OCTET STRING may contain arbitrary binary data, in which each octet may take any value from 0 to 255 (decimal).

3.3. Physical Addresses

As a further, textual convention in the MIB, the datatype

```
PhysAddress ::=
    OCTET STRING
```

is introduced to represent media- or physical-level addresses.

The following objects are now defined in terms of PhysAddress:

```
ifPhysAddress
atPhysAddress
ipNetToMediaPhysAddress
```

It should be noted that this change has no effect on either the syntax nor semantics of these objects. The use of the PhysAddress notation is merely an artifact of the explanatory method used in MIB-II and future MIBs.

3.4. The System Group

Four new objects are added to this group:

sysContact
sysName
sysLocation
sysServices

These provide contact, administrative, location, and service information regarding the managed node.

3.5. The Interfaces Group

The definition of the ifNumber object was incorrect, as it required all interfaces to support IP. (For example, devices without IP, such as MAC-layer bridges, could not be managed if this definition was strictly followed.) The description of the ifNumber object is changed accordingly.

The ifTable object was mistaken marked as read-write, it has been (correctly) re-designated as not-accessible. In addition, several new values have been added to the ifType column in the ifTable object:

ppp(23)
softwareLoopback(24)
eon(25)
ethernet-3Mbit(26)
nsip(27)
slip(28)
ultra(29)
ds3(30)
sip(31)
frame-relay(32)

Finally, a new column has been added to the ifTable object:

ifSpecific

which provides information about information specific to the media being used to realize the interface.

3.6. The Address Translation Group

In MIB-I this group contained a table which permitted mappings from network addresses (e.g., IP addresses) to physical addresses (e.g., MAC addresses). Experience has shown that efficient implementations of this table make two assumptions: a single network protocol environment, and mappings occur only from network address to physical address.

The need to support multi-protocol nodes (e.g., those with both the IP and CLNP active), and the need to support the inverse mapping (e.g., for ES-IS), have invalidated both of these assumptions. As such, the atTable object is declared deprecated.

In order to meet both the multi-protocol and inverse mapping requirements, MIB-II and its successors will allocate up to two address translation tables inside each network protocol group. That is, the IP group will contain one address translation table, for going from IP addresses to physical addresses. Similarly, when a document defining MIB objects for the CLNP is produced (e.g., [7]), it will contain two tables, for mappings in both directions, as this is required for full functionality.

It should be noted that the choice of two tables (one for each direction of mapping) provides for ease of implementation in many cases, and does not introduce undue burden on implementations which realize the address translation abstraction through a single internal table.

3.7. The IP Group

The access attribute of the variable ipForwarding has been changed from read-only to read-write.

In addition, there is a new column to the ipAddrTable object,

    ipAdEntReasmMaxSize

which keeps track of the largest IP datagram that can be re-assembled on a particular interface.

The descriptor of the ipRoutingTable object has been changed to ipRouteTable for consistency with the other IP routing objects. There are also three new columns in the ipRouteTable object,

    ipRouteMask
    ipRouteMetric5
    ipRouteInfo

the first is used for IP routing subsystems that support arbitrary subnet masks, and the latter two are IP routing protocol-specific.

Two new objects are added to the IP group:

    ipNetToMediaTable
    ipRoutingDiscards

the first is the address translation table for the IP group (providing identical functionality to the now deprecated atTable in the address translation group), and the latter provides information when routes are lost due to a lack of buffer space.

3.8. The ICMP Group

There are no changes to this group.

3.9. The TCP Group

Two new variables are added:

    tcpInErrs
    tcpOutRsts

which keep track of the number of incoming TCP segments in error and the number of resets generated by a TCP.

3.10. The UDP Group

A new table:

    udpTable

is added.

3.11. The EGP Group

Experience has indicated a need for additional objects that are useful in EGP monitoring. In addition to making several additions to the egpNeighborTable object, i.e.,

    egpNeighAs
    egpNeighInMsgs
    egpNeighInErrs
    egpNeighOutMsgs
    egpNeighOutErrs
    egpNeighInErrMsgs
    egpNeighOutErrMsgs

```
egpNeighStateUps
egpNeighStateDowns
egpNeighIntervalHello
egpNeighIntervalPoll
egpNeighMode
egpNeighEventTrigger
```

a new variable is added:

```
egpAs
```

which gives the autonomous system associated with this EGP entity.

3.12. The Transmission Group

MIB-I was lacking in that it did not distinguish between different types of transmission media. A new group, the Transmission group, is allocated for this purpose:

```
transmission OBJECT IDENTIFIER ::= { mib-2 10 }
```

When Internet-standard definitions for managing transmission media are defined, the transmission group is used to provide a prefix for the names of those objects.

Typically, such definitions reside in the experimental portion of the MIB until they are "proven", then as a part of the Internet standardization process, the definitions are accordingly elevated and a new object identifier, under the transmission group is defined. By convention, the name assigned is:

```
type OBJECT IDENTIFIER ::= { transmission number }
```

where "type" is the symbolic value used for the media in the ifType column of the ifTable object, and "number" is the actual integer value corresponding to the symbol.

3.13. The SNMP Group

The application-oriented working groups of the IETF have been tasked to be receptive towards defining MIB variables specific to their respective applications.

For the SNMP, it is useful to have statistical information. A new group, the SNMP group, is allocated for this purpose:

```
snmp   OBJECT IDENTIFIER ::= { mib-2 11 }
```

3.14. Changes from RFC 1158

Features of this MIB include:

(1) The managed objects in this document have been defined using the conventions defined in the Internet-standard SMI, as amended by the extensions specified in [14]. It must be emphasized that definitions made using these extensions are semantically identically to those in RFC 1158.

(2) The PhysAddress textual convention has been introduced to represent media addresses.

(3) The ACCESS clause of sysLocation is now read-write.

(4) The definition of sysServices has been clarified.

(5) New ifType values (29-32) have been defined. In addition, the textual-descriptor for the DS1 and E1 interface types has been corrected.

(6) The definition of ipForwarding has been clarified.

(7) The definition of ipRouteType has been clarified.

(8) The ipRouteMetric5 and ipRouteInfo objects have been defined.

(9) The ACCESS clause of tcpConnState is now read-write, to support deletion of the TCB associated with a TCP connection. The definition of this object has been clarified to explain this usage.

(10) The definition of egpNeighEventTrigger has been clarified.

(11) The definition of several of the variables in the new snmp group have been clarified. In addition, the snmpInBadTypes and snmpOutReadOnlys objects are no longer present. (However, the object identifiers associated with those objects are reserved to prevent future use.)

(12) The definition of snmpInReadOnlys has been clarified.

(13) The textual descriptor of the snmpEnableAuthTraps has been changed to snmpEnableAuthenTraps, and the definition has been clarified.

(14) The ipRoutingDiscards object was added.

(15) The optional use of an implementation-dependent, small positive integer was disallowed when identifying instances of the IP address and routing tables.

4. Objects

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the subset of Abstract Syntax Notation One (ASN.1) [8] defined in the SMI. In particular, each object has a name, a syntax, and an encoding. The name is an object identifier, an administratively assigned name, which specifies an object type. The object type together with an object instance serves to uniquely identify a specific instantiation of the object. For human convenience, we often use a textual string, termed the OBJECT DESCRIPTOR, to also refer to the object type.

The syntax of an object type defines the abstract data structure corresponding to that object type. The ASN.1 language is used for this purpose. However, the SMI [12] purposely restricts the ASN.1 constructs which may be used. These restrictions are explicitly made for simplicity.

The encoding of an object type is simply how that object type is represented using the object type's syntax. Implicitly tied to the notion of an object type's syntax and encoding is how the object type is represented when being transmitted on the network.

The SMI specifies the use of the basic encoding rules of ASN.1 [9], subject to the additional requirements imposed by the SNMP.

4.1. Format of Definitions

Section 6 contains contains the specification of all object types contained in this MIB module. The object types are defined using the conventions defined in the SMI, as amended by the extensions specified in [14].

5. Overview

Consistent with the IAB directive to produce simple, workable systems in the short-term, the list of managed objects defined here, has been derived by taking only those elements which are considered essential.

This approach of taking only the essential objects is NOT restrictive, since the SMI defined in the companion memo provides three extensibility mechanisms: one, the addition of new standard objects through the definitions of new versions of the MIB; two, the addition of widely-available but non-standard objects through the experimental subtree; and three, the addition of private objects through the enterprises subtree. Such additional objects can not only be used for vendor-specific elements, but also for experimentation as required to further the knowledge of which other objects are essential.

The design of MIB-II is heavily influenced by the first extensibility mechanism. Several new variables have been added based on operational experience and need. Based on this, the criteria for including an object in MIB-II are remarkably similar to the MIB-I criteria:

(1) An object needed to be essential for either fault or configuration management.

(2) Only weak control objects were permitted (by weak, it is meant that tampering with them can do only limited damage). This criterion reflects the fact that the current management protocols are not sufficiently secure to do more powerful control operations.

(3) Evidence of current use and utility was required.

(4) In MIB-I, an attempt was made to limit the number of objects to about 100 to make it easier for vendors to fully instrument their software. In MIB-II, this limit was raised given the wide technological base now implementing MIB-I.

(5) To avoid redundant variables, it was required that no object be included that can be derived from others in the MIB.

(6) Implementation specific objects (e.g., for BSD UNIX) were excluded.

(7) It was agreed to avoid heavily instrumenting critical sections of code. The general guideline was one counter per critical section per layer.

MIB-II, like its predecessor, the Internet-standard MIB, contains only essential elements. There is no need to allow individual objects to be optional. Rather, the objects are arranged into the following groups:

- System
- Interfaces
- Address Translation (deprecated)
- IP
- ICMP
- TCP
- UDP
- EGP
- Transmission
- SNMP

These groups are the basic unit of conformance: This method is as follows: if the semantics of a group is applicable to an implementation, then it must implement all objects in that group. For example, an implementation must implement the EGP group if and only if it implements the EGP.

There are two reasons for defining these groups: to provide a means of assigning object identifiers; and, to provide a method for implementations of managed agents to know which objects they must implement.

6. Definitions

```
RFC1213-MIB DEFINITIONS ::= BEGIN

IMPORTS
        mgmt, NetworkAddress, IpAddress, Counter, Gauge,
                TimeTicks
            FROM RFC1155-SMI
        OBJECT-TYPE
                FROM RFC-1212;

--  This MIB module uses the extended OBJECT-TYPE macro as
--  defined in [14];


--  MIB-II (same prefix as MIB-I)

mib-2      OBJECT IDENTIFIER ::= { mgmt 1 }

-- textual conventions

DisplayString ::=
    OCTET STRING
-- This data type is used to model textual information taken
-- from the NVT ASCII character set.  By convention, objects
-- with this syntax are declared as having
```

```
--
--      SIZE (0..255)

PhysAddress ::=
    OCTET STRING
-- This data type is used to model media addresses.  For many
-- types of media, this will be in a binary representation.
-- For example, an ethernet address would be represented as
-- a string of 6 octets.


-- groups in MIB-II

system       OBJECT IDENTIFIER ::= { mib-2 1 }

interfaces   OBJECT IDENTIFIER ::= { mib-2 2 }

at           OBJECT IDENTIFIER ::= { mib-2 3 }

ip           OBJECT IDENTIFIER ::= { mib-2 4 }

icmp         OBJECT IDENTIFIER ::= { mib-2 5 }

tcp          OBJECT IDENTIFIER ::= { mib-2 6 }

udp          OBJECT IDENTIFIER ::= { mib-2 7 }

egp          OBJECT IDENTIFIER ::= { mib-2 8 }

-- historical (some say hysterical)
-- cmot      OBJECT IDENTIFIER ::= { mib-2 9 }

transmission OBJECT IDENTIFIER ::= { mib-2 10 }

snmp         OBJECT IDENTIFIER ::= { mib-2 11 }


-- the System group

-- Implementation of the System group is mandatory for all
-- systems.  If an agent is not configured to have a value
-- for any of these variables, a string of length 0 is
-- returned.

sysDescr OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-only
    STATUS  mandatory
```

```
    DESCRIPTION
            "A textual description of the entity.  This value
            should include the full name and version
            identification of the system's hardware type,
            software operating-system, and networking
            software.  It is mandatory that this only contain
            printable ASCII characters."
    ::= { system 1 }

sysObjectID OBJECT-TYPE
    SYNTAX  OBJECT IDENTIFIER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The vendor's authoritative identification of the
            network management subsystem contained in the
            entity.  This value is allocated within the SMI
            enterprises subtree (1.3.6.1.4.1) and provides an
            easy and unambiguous means for determining `what
            kind of box' is being managed.  For example, if
            vendor `Flintstones, Inc.' was assigned the
            subtree 1.3.6.1.4.1.4242, it could assign the
            identifier 1.3.6.1.4.1.4242.1.1 to its `Fred
            Router'."
    ::= { system 2 }

sysUpTime OBJECT-TYPE
    SYNTAX  TimeTicks
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The time (in hundredths of a second) since the
            network management portion of the system was last
            re-initialized."
    ::= { system 3 }

sysContact OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The textual identification of the contact person
            for this managed node, together with information
            on how to contact this person."
    ::= { system 4 }

sysName OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
```

```
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "An administratively-assigned name for this
            managed node.  By convention, this is the node's
            fully-qualified domain name."
    ::= { system 5 }

sysLocation OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The physical location of this node (e.g.,
            `telephone closet, 3rd floor')."
    ::= { system 6 }

sysServices OBJECT-TYPE
    SYNTAX  INTEGER (0..127)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "A value which indicates the set of services that
            this entity primarily offers.

            The value is a sum.  This sum initially takes the
            value zero, Then, for each layer, L, in the range
            1 through 7, that this node performs transactions
            for, 2 raised to (L - 1) is added to the sum.  For
            example, a node which performs primarily routing
            functions would have a value of 4 (2^(3-1)).  In
            contrast, a node which is a host offering
            application services would have a value of 72
            (2^(4-1) + 2^(7-1)).  Note that in the context of
            the Internet suite of protocols, values should be
            calculated accordingly:

                 layer  functionality
                     1  physical (e.g., repeaters)
                     2  datalink/subnetwork (e.g., bridges)
                     3  internet (e.g., IP gateways)
                     4  end-to-end  (e.g., IP hosts)
                     7  applications (e.g., mail relays)

            For systems including OSI protocols, layers 5 and
            6 may also be counted."
    ::= { system 7 }
```

```
-- the Interfaces group

-- Implementation of the Interfaces group is mandatory for
-- all systems.

ifNumber OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of network interfaces (regardless of
            their current state) present on this system."
    ::= { interfaces 1 }


-- the Interfaces table

-- The Interfaces table contains information on the entity's
-- interfaces.  Each interface is thought of as being
-- attached to a 'subnetwork'.  Note that this term should
-- not be confused with 'subnet' which refers to an
-- addressing partitioning scheme used in the Internet suite
-- of protocols.

ifTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF IfEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "A list of interface entries.  The number of
            entries is given by the value of ifNumber."
    ::= { interfaces 2 }

ifEntry OBJECT-TYPE
    SYNTAX  IfEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "An interface entry containing objects at the
            subnetwork layer and below for a particular
            interface."
    INDEX   { ifIndex }
    ::= { ifTable 1 }

IfEntry ::=
    SEQUENCE {
        ifIndex
            INTEGER,
```

```
                ifDescr
                    DisplayString,
                ifType
                    INTEGER,
                ifMtu
                    INTEGER,
                ifSpeed
                    Gauge,
                ifPhysAddress
                    PhysAddress,
                ifAdminStatus
                    INTEGER,
                ifOperStatus
                    INTEGER,
                ifLastChange
                    TimeTicks,
                ifInOctets
                    Counter,
                ifInUcastPkts
                    Counter,
                ifInNUcastPkts
                    Counter,
                ifInDiscards
                    Counter,
                ifInErrors
                    Counter,
                ifInUnknownProtos
                    Counter,
                ifOutOctets
                    Counter,
                ifOutUcastPkts
                    Counter,
                ifOutNUcastPkts
                    Counter,
                ifOutDiscards
                    Counter,
                ifOutErrors
                    Counter,
                ifOutQLen
                    Gauge,
                ifSpecific
                    OBJECT IDENTIFIER
            }

        ifIndex OBJECT-TYPE
            SYNTAX  INTEGER
            ACCESS  read-only
            STATUS  mandatory
```

```
        DESCRIPTION
                "A unique value for each interface.  Its value
                ranges between 1 and the value of ifNumber.  The
                value for each interface must remain constant at
                least from one re-initialization of the entity's
                network management system to the next re-
                initialization."
        ::= { ifEntry 1 }

    ifDescr OBJECT-TYPE
        SYNTAX  DisplayString (SIZE (0..255))
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "A textual string containing information about the
                interface.  This string should include the name of
                the manufacturer, the product name and the version
                of the hardware interface."
        ::= { ifEntry 2 }

    ifType OBJECT-TYPE
        SYNTAX  INTEGER {
                    other(1),          -- none of the following
                    regular1822(2),
                    hdh1822(3),
                    ddn-x25(4),
                    rfc877-x25(5),
                    ethernet-csmacd(6),
                    iso88023-csmacd(7),
                    iso88024-tokenBus(8),
                    iso88025-tokenRing(9),
                    iso88026-man(10),
                    starLan(11),
                    proteon-10Mbit(12),
                    proteon-80Mbit(13),
                    hyperchannel(14),
                    fddi(15),
                    lapb(16),
                    sdlc(17),
                    ds1(18),           -- T-1
                    e1(19),            -- european equiv. of T-1
                    basicISDN(20),
                    primaryISDN(21),   -- proprietary serial
                    propPointToPointSerial(22),
                    ppp(23),
                    softwareLoopback(24),
                    eon(25),            -- CLNP over IP [11]
                    ethernet-3Mbit(26),
```

```
                    nsip(27),           -- XNS over IP
                    slip(28),           -- generic SLIP
                    ultra(29),          -- ULTRA technologies
                    ds3(30),            -- T-3
                    sip(31),            -- SMDS
                    frame-relay(32)
                }
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                    "The type of interface, distinguished according to
                    the physical/link protocol(s) immediately `below'
                    the network layer in the protocol stack."
            ::= { ifEntry 3 }

        ifMtu OBJECT-TYPE
            SYNTAX  INTEGER
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                    "The size of the largest datagram which can be
                    sent/received on the interface, specified in
                    octets.  For interfaces that are used for
                    transmitting network datagrams, this is the size
                    of the largest network datagram that can be sent
                    on the interface."
            ::= { ifEntry 4 }

        ifSpeed OBJECT-TYPE
            SYNTAX  Gauge
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                    "An estimate of the interface's current bandwidth
                    in bits per second.  For interfaces which do not
                    vary in bandwidth or for those where no accurate
                    estimation can be made, this object should contain
                    the nominal bandwidth."
            ::= { ifEntry 5 }

        ifPhysAddress OBJECT-TYPE
            SYNTAX  PhysAddress
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                    "The interface's address at the protocol layer
                    immediately `below' the network layer in the
                    protocol stack.  For interfaces which do not have
```

```
                such an address (e.g., a serial line), this object
                should contain an octet string of zero length."
        ::= { ifEntry 6 }

    ifAdminStatus OBJECT-TYPE
        SYNTAX  INTEGER {
                    up(1),       -- ready to pass packets
                    down(2),
                    testing(3)   -- in some test mode
                }
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "The desired state of the interface.  The
                testing(3) state indicates that no operational
                packets can be passed."
        ::= { ifEntry 7 }

    ifOperStatus OBJECT-TYPE
        SYNTAX  INTEGER {
                    up(1),       -- ready to pass packets
                    down(2),
                    testing(3)   -- in some test mode
                }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The current operational state of the interface.
                The testing(3) state indicates that no operational
                packets can be passed."
        ::= { ifEntry 8 }

    ifLastChange OBJECT-TYPE
        SYNTAX  TimeTicks
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The value of sysUpTime at the time the interface
                entered its current operational state.  If the
                current state was entered prior to the last re-
                initialization of the local network management
                subsystem, then this object contains a zero
                value."
        ::= { ifEntry 9 }

    ifInOctets OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
```

```
    STATUS  mandatory
    DESCRIPTION
            "The total number of octets received on the
            interface, including framing characters."
    ::= { ifEntry 10 }

ifInUcastPkts OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of subnetwork-unicast packets
            delivered to a higher-layer protocol."
    ::= { ifEntry 11 }

ifInNUcastPkts OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of non-unicast (i.e., subnetwork-
            broadcast or subnetwork-multicast) packets
            delivered to a higher-layer protocol."
    ::= { ifEntry 12 }

ifInDiscards OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of inbound packets which were chosen
            to be discarded even though no errors had been
            detected to prevent their being deliverable to a
            higher-layer protocol.  One possible reason for
            discarding such a packet could be to free up
            buffer space."
    ::= { ifEntry 13 }

ifInErrors OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of inbound packets that contained
            errors preventing them from being deliverable to a
            higher-layer protocol."
    ::= { ifEntry 14 }
```

```
ifInUnknownProtos OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of packets received via the interface
            which were discarded because of an unknown or
            unsupported protocol."
    ::= { ifEntry 15 }

ifOutOctets OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of octets transmitted out of the
            interface, including framing characters."
    ::= { ifEntry 16 }

ifOutUcastPkts OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of packets that higher-level
            protocols requested be transmitted to a
            subnetwork-unicast address, including those that
            were discarded or not sent."
    ::= { ifEntry 17 }

ifOutNUcastPkts OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of packets that higher-level
            protocols requested be transmitted to a non-
            unicast (i.e., a subnetwork-broadcast or
            subnetwork-multicast) address, including those
            that were discarded or not sent."
    ::= { ifEntry 18 }

ifOutDiscards OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of outbound packets which were chosen
```

```
                    to be discarded even though no errors had been
                    detected to prevent their being transmitted.  One
                    possible reason for discarding such a packet could
                    be to free up buffer space."
            ::= { ifEntry 19 }

        ifOutErrors OBJECT-TYPE
            SYNTAX  Counter
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                    "The number of outbound packets that could not be
                    transmitted because of errors."
            ::= { ifEntry 20 }

        ifOutQLen OBJECT-TYPE
            SYNTAX  Gauge
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                    "The length of the output packet queue (in
                    packets)."
            ::= { ifEntry 21 }

        ifSpecific OBJECT-TYPE
            SYNTAX  OBJECT IDENTIFIER
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                    "A reference to MIB definitions specific to the
                    particular media being used to realize the
                    interface.  For example, if the interface is
                    realized by an ethernet, then the value of this
                    object refers to a document defining objects
                    specific to ethernet.  If this information is not
                    present, its value should be set to the OBJECT
                    IDENTIFIER { 0 0 }, which is a syntatically valid
                    object identifier, and any conformant
                    implementation of ASN.1 and BER must be able to
                    generate and recognize this value."
            ::= { ifEntry 22 }


        -- the Address Translation group

        -- Implementation of the Address Translation group is
        -- mandatory for all systems.  Note however that this group
        -- is deprecated by MIB-II. That is, it is being included
```

```
-- solely for compatibility with MIB-I nodes, and will most
-- likely be excluded from MIB-III nodes.  From MIB-II and
-- onwards, each network protocol group contains its own
-- address translation tables.

-- The Address Translation group contains one table which is
-- the union across all interfaces of the translation tables
-- for converting a NetworkAddress (e.g., an IP address) into
-- a subnetwork-specific address.  For lack of a better term,
-- this document refers to such a subnetwork-specific address
-- as a `physical' address.

-- Examples of such translation tables are: for broadcast
-- media where ARP is in use, the translation table is
-- equivalent to the ARP cache; or, on an X.25 network where
-- non-algorithmic translation to X.121 addresses is
-- required, the translation table contains the
-- NetworkAddress to X.121 address equivalences.

atTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF AtEntry
    ACCESS  not-accessible
    STATUS  deprecated
    DESCRIPTION
            "The Address Translation tables contain the
            NetworkAddress to `physical' address equivalences.
            Some interfaces do not use translation tables for
            determining address equivalences (e.g., DDN-X.25
            has an algorithmic method); if all interfaces are
            of this type, then the Address Translation table
            is empty, i.e., has zero entries."
    ::= { at 1 }

atEntry OBJECT-TYPE
    SYNTAX  AtEntry
    ACCESS  not-accessible
    STATUS  deprecated
    DESCRIPTION
            "Each entry contains one NetworkAddress to
            `physical' address equivalence."
    INDEX   { atIfIndex,
              atNetAddress }
    ::= { atTable 1 }

AtEntry ::=
    SEQUENCE {
        atIfIndex
            INTEGER,
```

```
            atPhysAddress
                PhysAddress,
            atNetAddress
                NetworkAddress
        }

    atIfIndex OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-write
        STATUS  deprecated
        DESCRIPTION
                "The interface on which this entry's equivalence
                is effective.  The interface identified by a
                particular value of this index is the same
                interface as identified by the same value of
                ifIndex."
        ::= { atEntry 1 }

    atPhysAddress OBJECT-TYPE
        SYNTAX  PhysAddress
        ACCESS  read-write
        STATUS  deprecated
        DESCRIPTION
                "The media-dependent `physical' address.

                Setting this object to a null string (one of zero
                length) has the effect of invaliding the
                corresponding entry in the atTable object.  That
                is, it effectively dissasociates the interface
                identified with said entry from the mapping
                identified with said entry.  It is an
                implementation-specific matter as to whether the
                agent removes an invalidated entry from the table.
                Accordingly, management stations must be prepared
                to receive tabular information from agents that
                corresponds to entries not currently in use.
                Proper interpretation of such entries requires
                examination of the relevant atPhysAddress object."
        ::= { atEntry 2 }

    atNetAddress OBJECT-TYPE
        SYNTAX  NetworkAddress
        ACCESS  read-write
        STATUS  deprecated
        DESCRIPTION
                "The NetworkAddress (e.g., the IP address)
                corresponding to the media-dependent `physical'
                address."
```

```
    ::= { atEntry 3 }


-- the IP group

-- Implementation of the IP group is mandatory for all
-- systems.

ipForwarding OBJECT-TYPE
    SYNTAX  INTEGER {
                forwarding(1),     -- acting as a gateway
                not-forwarding(2)  -- NOT acting as a gateway
            }
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The indication of whether this entity is acting
            as an IP gateway in respect to the forwarding of
            datagrams received by, but not addressed to, this
            entity.  IP gateways forward datagrams.  IP hosts
            do not (except those source-routed via the host).

            Note that for some managed nodes, this object may
            take on only a subset of the values possible.
            Accordingly, it is appropriate for an agent to
            return a 'badValue' response if a management
            station attempts to change this object to an
            inappropriate value."
    ::= { ip 1 }

ipDefaultTTL OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The default value inserted into the Time-To-Live
            field of the IP header of datagrams originated at
            this entity, whenever a TTL value is not supplied
            by the transport layer protocol."
    ::= { ip 2 }

ipInReceives OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of input datagrams received from
            interfaces, including those received in error."
```

```
    ::= { ip 3 }

ipInHdrErrors OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of input datagrams discarded due to
            errors in their IP headers, including bad
            checksums, version number mismatch, other format
            errors, time-to-live exceeded, errors discovered
            in processing their IP options, etc."
    ::= { ip 4 }

ipInAddrErrors OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of input datagrams discarded because
            the IP address in their IP header's destination
            field was not a valid address to be received at
            this entity.  This count includes invalid
            addresses (e.g., 0.0.0.0) and addresses of
            unsupported Classes (e.g., Class E).  For entities
            which are not IP Gateways and therefore do not
            forward datagrams, this counter includes datagrams
            discarded because the destination address was not
            a local address."
    ::= { ip 5 }

ipForwDatagrams OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of input datagrams for which this
            entity was not their final IP destination, as a
            result of which an attempt was made to find a
            route to forward them to that final destination.
            In entities which do not act as IP Gateways, this
            counter will include only those packets which were
            Source-Routed via this entity, and the Source-
            Route option processing was successful."
    ::= { ip 6 }

ipInUnknownProtos OBJECT-TYPE
    SYNTAX  Counter
```

```
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of locally-addressed datagrams
            received successfully but discarded because of an
            unknown or unsupported protocol."
    ::= { ip 7 }

ipInDiscards OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of input IP datagrams for which no
            problems were encountered to prevent their
            continued processing, but which were discarded
            (e.g., for lack of buffer space).  Note that this
            counter does not include any datagrams discarded
            while awaiting re-assembly."
    ::= { ip 8 }

ipInDelivers OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of input datagrams successfully
            delivered to IP user-protocols (including ICMP)."
    ::= { ip 9 }

ipOutRequests OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of IP datagrams which local IP
            user-protocols (including ICMP) supplied to IP in
            requests for transmission.  Note that this counter
            does not include any datagrams counted in
            ipForwDatagrams."
    ::= { ip 10 }

ipOutDiscards OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of output IP datagrams for which no
```

```
                problem was encountered to prevent their
                transmission to their destination, but which were
                discarded (e.g., for lack of buffer space).  Note
                that this counter would include datagrams counted
                in ipForwDatagrams if any such packets met this
                (discretionary) discard criterion."
        ::= { ip 11 }

    ipOutNoRoutes OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of IP datagrams discarded because no
                route could be found to transmit them to their
                destination.  Note that this counter includes any
                packets counted in ipForwDatagrams which meet this
                `no-route' criterion.  Note that this includes any
                datagarms which a host cannot route because all of
                its default gateways are down."
        ::= { ip 12 }

    ipReasmTimeout OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The maximum number of seconds which received
                fragments are held while they are awaiting
                reassembly at this entity."
        ::= { ip 13 }

    ipReasmReqds OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of IP fragments received which needed
                to be reassembled at this entity."
        ::= { ip 14 }

    ipReasmOKs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of IP datagrams successfully re-
                assembled."
```

```
    ::= { ip 15 }

ipReasmFails OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of failures detected by the IP re-
            assembly algorithm (for whatever reason: timed
            out, errors, etc).  Note that this is not
            necessarily a count of discarded IP fragments
            since some algorithms (notably the algorithm in
            RFC 815) can lose track of the number of fragments
            by combining them as they are received."
    ::= { ip 16 }

ipFragOKs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of IP datagrams that have been
            successfully fragmented at this entity."
    ::= { ip 17 }

ipFragFails OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of IP datagrams that have been
            discarded because they needed to be fragmented at
            this entity but could not be, e.g., because their
            Don't Fragment flag was set."
    ::= { ip 18 }

ipFragCreates OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of IP datagram fragments that have
            been generated as a result of fragmentation at
            this entity."
    ::= { ip 19 }
```

```
-- the IP address table

-- The IP address table contains this entity's IP addressing
-- information.

ipAddrTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF IpAddrEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "The table of addressing information relevant to
            this entity's IP addresses."
    ::= { ip 20 }

ipAddrEntry OBJECT-TYPE
    SYNTAX  IpAddrEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "The addressing information for one of this
            entity's IP addresses."
    INDEX   { ipAdEntAddr }
    ::= { ipAddrTable 1 }

IpAddrEntry ::=
    SEQUENCE {
        ipAdEntAddr
            IpAddress,
        ipAdEntIfIndex
            INTEGER,
        ipAdEntNetMask
            IpAddress,
        ipAdEntBcastAddr
            INTEGER,
        ipAdEntReasmMaxSize
            INTEGER (0..65535)
    }

ipAdEntAddr OBJECT-TYPE
    SYNTAX  IpAddress
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The IP address to which this entry's addressing
            information pertains."
    ::= { ipAddrEntry 1 }
```

```
ipAdEntIfIndex OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The index value which uniquely identifies the
            interface to which this entry is applicable.  The
            interface identified by a particular value of this
            index is the same interface as identified by the
            same value of ifIndex."
    ::= { ipAddrEntry 2 }

ipAdEntNetMask OBJECT-TYPE
    SYNTAX  IpAddress
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The subnet mask associated with the IP address of
            this entry.  The value of the mask is an IP
            address with all the network bits set to 1 and all
            the hosts bits set to 0."
    ::= { ipAddrEntry 3 }

ipAdEntBcastAddr OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The value of the least-significant bit in the IP
            broadcast address used for sending datagrams on
            the (logical) interface associated with the IP
            address of this entry.  For example, when the
            Internet standard all-ones broadcast address is
            used, the value will be 1.  This value applies to
            both the subnet and network broadcasts addresses
            used by the entity on this (logical) interface."
    ::= { ipAddrEntry 4 }

ipAdEntReasmMaxSize OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The size of the largest IP datagram which this
            entity can re-assemble from incoming IP fragmented
            datagrams received on this interface."
    ::= { ipAddrEntry 5 }
```

```
-- the IP routing table

-- The IP routing table contains an entry for each route
-- presently known to this entity.

ipRouteTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF IpRouteEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "This entity's IP Routing table."
    ::= { ip 21 }

ipRouteEntry OBJECT-TYPE
    SYNTAX  IpRouteEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "A route to a particular destination."
    INDEX   { ipRouteDest }
    ::= { ipRouteTable 1 }

IpRouteEntry ::=
    SEQUENCE {
        ipRouteDest
            IpAddress,
        ipRouteIfIndex
            INTEGER,
        ipRouteMetric1
            INTEGER,
        ipRouteMetric2
            INTEGER,
        ipRouteMetric3
            INTEGER,
        ipRouteMetric4
            INTEGER,
        ipRouteNextHop
            IpAddress,
        ipRouteType
            INTEGER,
        ipRouteProto
            INTEGER,
        ipRouteAge
            INTEGER,
        ipRouteMask
            IpAddress,
        ipRouteMetric5
            INTEGER,
```

```
            ipRouteInfo
                OBJECT IDENTIFIER
        }

    ipRouteDest OBJECT-TYPE
        SYNTAX  IpAddress
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "The destination IP address of this route.  An
                entry with a value of 0.0.0.0 is considered a
                default route.  Multiple routes to a single
                destination can appear in the table, but access to
                such multiple entries is dependent on the table-
                access mechanisms defined by the network
                management protocol in use."
        ::= { ipRouteEntry 1 }

    ipRouteIfIndex OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "The index value which uniquely identifies the
                local interface through which the next hop of this
                route should be reached.  The interface identified
                by a particular value of this index is the same
                interface as identified by the same value of
                ifIndex."
        ::= { ipRouteEntry 2 }

    ipRouteMetric1 OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "The primary routing metric for this route.  The
                semantics of this metric are determined by the
                routing-protocol specified in the route's
                ipRouteProto value.  If this metric is not used,
                its value should be set to -1."
        ::= { ipRouteEntry 3 }

    ipRouteMetric2 OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
```

```
                "An alternate routing metric for this route.  The
                semantics of this metric are determined by the
                routing-protocol specified in the route's
                ipRouteProto value.  If this metric is not used,
                its value should be set to -1."
        ::= { ipRouteEntry 4 }

    ipRouteMetric3 OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "An alternate routing metric for this route.  The
                semantics of this metric are determined by the
                routing-protocol specified in the route's
                ipRouteProto value.  If this metric is not used,
                its value should be set to -1."
        ::= { ipRouteEntry 5 }

    ipRouteMetric4 OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "An alternate routing metric for this route.  The
                semantics of this metric are determined by the
                routing-protocol specified in the route's
                ipRouteProto value.  If this metric is not used,
                its value should be set to -1."
        ::= { ipRouteEntry 6 }

    ipRouteNextHop OBJECT-TYPE
        SYNTAX  IpAddress
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "The IP address of the next hop of this route.
                (In the case of a route bound to an interface
                which is realized via a broadcast media, the value
                of this field is the agent's IP address on that
                interface.)"
        ::= { ipRouteEntry 7 }

    ipRouteType OBJECT-TYPE
        SYNTAX  INTEGER {
                    other(1),        -- none of the following

                    invalid(2),      -- an invalidated route
```

```
                                  -- route to directly
                direct(3),        -- connected (sub-)network

                                  -- route to a non-local
                indirect(4)       -- host/network/sub-network
            }
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "The type of route.  Note that the values
                direct(3) and indirect(4) refer to the notion of
                direct and indirect routing in the IP
                architecture.

                Setting this object to the value invalid(2) has
                the effect of invalidating the corresponding entry
                in the ipRouteTable object.  That is, it
                effectively dissasociates the destination
                identified with said entry from the route
                identified with said entry.  It is an
                implementation-specific matter as to whether the
                agent removes an invalidated entry from the table.
                Accordingly, management stations must be prepared
                to receive tabular information from agents that
                corresponds to entries not currently in use.
                Proper interpretation of such entries requires
                examination of the relevant ipRouteType object."
        ::= { ipRouteEntry 8 }

    ipRouteProto OBJECT-TYPE
        SYNTAX  INTEGER {
                    other(1),       -- none of the following

                                    -- non-protocol information,
                                    -- e.g., manually configured
                    local(2),       -- entries

                                    -- set via a network
                    netmgmt(3),     -- management protocol

                                    -- obtained via ICMP,
                    icmp(4),        -- e.g., Redirect

                                    -- the remaining values are
                                    -- all gateway routing
                                    -- protocols
                    egp(5),
                    ggp(6),
```

```
                    hello(7),
                    rip(8),
                    is-is(9),
                    es-is(10),
                    ciscoIgrp(11),
                    bbnSpfIgp(12),
                    ospf(13),
                    bgp(14)
                }
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                    "The routing mechanism via which this route was
                    learned.  Inclusion of values for gateway routing
                    protocols is not intended to imply that hosts
                    should support those protocols."
            ::= { ipRouteEntry 9 }

        ipRouteAge OBJECT-TYPE
            SYNTAX  INTEGER
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "The number of seconds since this route was last
                    updated or otherwise determined to be correct.
                    Note that no semantics of `too old' can be implied
                    except through knowledge of the routing protocol
                    by which the route was learned."
            ::= { ipRouteEntry 10 }

        ipRouteMask OBJECT-TYPE
            SYNTAX  IpAddress
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "Indicate the mask to be logical-ANDed with the
                    destination address before being compared to the
                    value in the ipRouteDest field.  For those systems
                    that do not support arbitrary subnet masks, an
                    agent constructs the value of the ipRouteMask by
                    determining whether the value of the correspondent
                    ipRouteDest field belong to a class-A, B, or C
                    network, and then using one of:

                         mask           network
                         255.0.0.0      class-A
                         255.255.0.0    class-B
                         255.255.255.0  class-C
```

```
                If the value of the ipRouteDest is 0.0.0.0 (a
                default route), then the mask value is also
                0.0.0.0.  It should be noted that all IP routing
                subsystems implicitly use this mechanism."
        ::= { ipRouteEntry 11 }

    ipRouteMetric5 OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "An alternate routing metric for this route.  The
                semantics of this metric are determined by the
                routing-protocol specified in the route's
                ipRouteProto value.  If this metric is not used,
                its value should be set to -1."
        ::= { ipRouteEntry 12 }

    ipRouteInfo OBJECT-TYPE
        SYNTAX  OBJECT IDENTIFIER
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "A reference to MIB definitions specific to the
                particular routing protocol which is responsible
                for this route, as determined by the value
                specified in the route's ipRouteProto value.  If
                this information is not present, its value should
                be set to the OBJECT IDENTIFIER { 0 0 }, which is
                a syntatically valid object identifier, and any
                conformant implementation of ASN.1 and BER must be
                able to generate and recognize this value."
        ::= { ipRouteEntry 13 }


    -- the IP Address Translation table

    -- The IP address translation table contain the IpAddress to
    -- 'physical' address equivalences.  Some interfaces do not
    -- use translation tables for determining address
    -- equivalences (e.g., DDN-X.25 has an algorithmic method);
    -- if all interfaces are of this type, then the Address
    -- Translation table is empty, i.e., has zero entries.

    ipNetToMediaTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF IpNetToMediaEntry
        ACCESS  not-accessible
        STATUS  mandatory
```

```
            DESCRIPTION
                    "The IP Address Translation table used for mapping
                    from IP addresses to physical addresses."
            ::= { ip 22 }

        ipNetToMediaEntry OBJECT-TYPE
            SYNTAX  IpNetToMediaEntry
            ACCESS  not-accessible
            STATUS  mandatory
            DESCRIPTION
                    "Each entry contains one IpAddress to 'physical'
                    address equivalence."
            INDEX   { ipNetToMediaIfIndex,
                      ipNetToMediaNetAddress }
            ::= { ipNetToMediaTable 1 }

        IpNetToMediaEntry ::=
            SEQUENCE {
                ipNetToMediaIfIndex
                    INTEGER,
                ipNetToMediaPhysAddress
                    PhysAddress,
                ipNetToMediaNetAddress
                    IpAddress,
                ipNetToMediaType
                    INTEGER
            }

        ipNetToMediaIfIndex OBJECT-TYPE
            SYNTAX  INTEGER
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "The interface on which this entry's equivalence
                    is effective.  The interface identified by a
                    particular value of this index is the same
                    interface as identified by the same value of
                    ifIndex."
            ::= { ipNetToMediaEntry 1 }

        ipNetToMediaPhysAddress OBJECT-TYPE
            SYNTAX  PhysAddress
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "The media-dependent 'physical' address."
            ::= { ipNetToMediaEntry 2 }
```

```
ipNetToMediaNetAddress OBJECT-TYPE
    SYNTAX  IpAddress
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The IpAddress corresponding to the media-
            dependent `physical' address."
    ::= { ipNetToMediaEntry 3 }

ipNetToMediaType OBJECT-TYPE
    SYNTAX  INTEGER {
                other(1),        -- none of the following
                invalid(2),      -- an invalidated mapping
                dynamic(3),
                static(4)
            }
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The type of mapping.

            Setting this object to the value invalid(2) has
            the effect of invalidating the corresponding entry
            in the ipNetToMediaTable.  That is, it effectively
            dissasociates the interface identified with said
            entry from the mapping identified with said entry.
            It is an implementation-specific matter as to
            whether the agent removes an invalidated entry
            from the table.  Accordingly, management stations
            must be prepared to receive tabular information
            from agents that corresponds to entries not
            currently in use.  Proper interpretation of such
            entries requires examination of the relevant
            ipNetToMediaType object."
    ::= { ipNetToMediaEntry 4 }


-- additional IP objects

ipRoutingDiscards OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of routing entries which were chosen
            to be discarded even though they are valid.  One
            possible reason for discarding such an entry could
            be to free-up buffer space for other routing
```

```
                entries."
        ::= { ip 23 }


    -- the ICMP group

    -- Implementation of the ICMP group is mandatory for all
    -- systems.

    icmpInMsgs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The total number of ICMP messages which the
                entity received.  Note that this counter includes
                all those counted by icmpInErrors."
        ::= { icmp 1 }

    icmpInErrors OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of ICMP messages which the entity
                received but determined as having ICMP-specific
                errors (bad ICMP checksums, bad length, etc.)."
        ::= { icmp 2 }

    icmpInDestUnreachs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of ICMP Destination Unreachable
                messages received."
        ::= { icmp 3 }

    icmpInTimeExcds OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of ICMP Time Exceeded messages
                received."
        ::= { icmp 4 }
```

```
icmpInParmProbs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of ICMP Parameter Problem messages
            received."
    ::= { icmp 5 }

icmpInSrcQuenchs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of ICMP Source Quench messages
            received."
    ::= { icmp 6 }

icmpInRedirects OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of ICMP Redirect messages received."
    ::= { icmp 7 }

icmpInEchos OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of ICMP Echo (request) messages
            received."
    ::= { icmp 8 }

icmpInEchoReps OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of ICMP Echo Reply messages received."
    ::= { icmp 9 }

icmpInTimestamps OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
```

```
                "The number of ICMP Timestamp (request) messages
                received."
        ::= { icmp 10 }

    icmpInTimestampReps OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of ICMP Timestamp Reply messages
                received."
        ::= { icmp 11 }

    icmpInAddrMasks OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of ICMP Address Mask Request messages
                received."
        ::= { icmp 12 }

    icmpInAddrMaskReps OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of ICMP Address Mask Reply messages
                received."
        ::= { icmp 13 }

    icmpOutMsgs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The total number of ICMP messages which this
                entity attempted to send.  Note that this counter
                includes all those counted by icmpOutErrors."
        ::= { icmp 14 }

    icmpOutErrors OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of ICMP messages which this entity did
                not send due to problems discovered within ICMP
```

```
                such as a lack of buffers.  This value should not
                include errors discovered outside the ICMP layer
                such as the inability of IP to route the resultant
                datagram.  In some implementations there may be no
                types of error which contribute to this counter's
                value."
        ::= { icmp 15 }

    icmpOutDestUnreachs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of ICMP Destination Unreachable
                messages sent."
        ::= { icmp 16 }

    icmpOutTimeExcds OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of ICMP Time Exceeded messages sent."
        ::= { icmp 17 }

    icmpOutParmProbs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of ICMP Parameter Problem messages
                sent."
        ::= { icmp 18 }

    icmpOutSrcQuenchs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of ICMP Source Quench messages sent."
        ::= { icmp 19 }

    icmpOutRedirects OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of ICMP Redirect messages sent.  For a
```

```
                host, this object will always be zero, since hosts
                do not send redirects."
        ::= { icmp 20 }

    icmpOutEchos OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of ICMP Echo (request) messages sent."
        ::= { icmp 21 }

    icmpOutEchoReps OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of ICMP Echo Reply messages sent."
        ::= { icmp 22 }

    icmpOutTimestamps OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of ICMP Timestamp (request) messages
                sent."
        ::= { icmp 23 }

    icmpOutTimestampReps OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of ICMP Timestamp Reply messages
                sent."
        ::= { icmp 24 }

    icmpOutAddrMasks OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of ICMP Address Mask Request messages
                sent."
        ::= { icmp 25 }
```

```
icmpOutAddrMaskReps OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of ICMP Address Mask Reply messages
            sent."
    ::= { icmp 26 }


-- the TCP group

-- Implementation of the TCP group is mandatory for all
-- systems that implement the TCP.

-- Note that instances of object types that represent
-- information about a particular TCP connection are
-- transient; they persist only as long as the connection
-- in question.

tcpRtoAlgorithm OBJECT-TYPE
    SYNTAX  INTEGER {
                other(1),    -- none of the following

                constant(2), -- a constant rto
                rsre(3),     -- MIL-STD-1778, Appendix B
                vanj(4)      -- Van Jacobson's algorithm [10]
            }
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The algorithm used to determine the timeout value
            used for retransmitting unacknowledged octets."
    ::= { tcp 1 }

tcpRtoMin OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The minimum value permitted by a TCP
            implementation for the retransmission timeout,
            measured in milliseconds.  More refined semantics
            for objects of this type depend upon the algorithm
            used to determine the retransmission timeout.  In
            particular, when the timeout algorithm is rsre(3),
            an object of this type has the semantics of the
            LBOUND quantity described in RFC 793."
```

```
    ::= { tcp 2 }


tcpRtoMax OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The maximum value permitted by a TCP
            implementation for the retransmission timeout,
            measured in milliseconds.  More refined semantics
            for objects of this type depend upon the algorithm
            used to determine the retransmission timeout.  In
            particular, when the timeout algorithm is rsre(3),
            an object of this type has the semantics of the
            UBOUND quantity described in RFC 793."
    ::= { tcp 3 }

tcpMaxConn OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The limit on the total number of TCP connections
            the entity can support.  In entities where the
            maximum number of connections is dynamic, this
            object should contain the value -1."
    ::= { tcp 4 }

tcpActiveOpens OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of times TCP connections have made a
            direct transition to the SYN-SENT state from the
            CLOSED state."
    ::= { tcp 5 }

tcpPassiveOpens OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of times TCP connections have made a
            direct transition to the SYN-RCVD state from the
            LISTEN state."
    ::= { tcp 6 }
```

```
tcpAttemptFails OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of times TCP connections have made a
            direct transition to the CLOSED state from either
            the SYN-SENT state or the SYN-RCVD state, plus the
            number of times TCP connections have made a direct
            transition to the LISTEN state from the SYN-RCVD
            state."
    ::= { tcp 7 }

tcpEstabResets OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of times TCP connections have made a
            direct transition to the CLOSED state from either
            the ESTABLISHED state or the CLOSE-WAIT state."
    ::= { tcp 8 }

tcpCurrEstab OBJECT-TYPE
    SYNTAX  Gauge
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of TCP connections for which the
            current state is either ESTABLISHED or CLOSE-
            WAIT."
    ::= { tcp 9 }

tcpInSegs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of segments received, including
            those received in error.  This count includes
            segments received on currently established
            connections."
    ::= { tcp 10 }

tcpOutSegs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
```

```
        DESCRIPTION
                "The total number of segments sent, including
                those on current connections but excluding those
                containing only retransmitted octets."
        ::= { tcp 11 }

    tcpRetransSegs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The total number of segments retransmitted - that
                is, the number of TCP segments transmitted
                containing one or more previously transmitted
                octets."
        ::= { tcp 12 }


    -- the TCP Connection table

    -- The TCP connection table contains information about this
    -- entity's existing TCP connections.

    tcpConnTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF TcpConnEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "A table containing TCP connection-specific
                information."
        ::= { tcp 13 }

    tcpConnEntry OBJECT-TYPE
        SYNTAX  TcpConnEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "Information about a particular current TCP
                connection.  An object of this type is transient,
                in that it ceases to exist when (or soon after)
                the connection makes the transition to the CLOSED
                state."
        INDEX   { tcpConnLocalAddress,
                  tcpConnLocalPort,
                  tcpConnRemAddress,
                  tcpConnRemPort }
        ::= { tcpConnTable 1 }
```

```
TcpConnEntry ::=
    SEQUENCE {
        tcpConnState
            INTEGER,
        tcpConnLocalAddress
            IpAddress,
        tcpConnLocalPort
            INTEGER (0..65535),
        tcpConnRemAddress
            IpAddress,
        tcpConnRemPort
            INTEGER (0..65535)
    }

tcpConnState OBJECT-TYPE
    SYNTAX  INTEGER {
                closed(1),
                listen(2),
                synSent(3),
                synReceived(4),
                established(5),
                finWait1(6),
                finWait2(7),
                closeWait(8),
                lastAck(9),
                closing(10),
                timeWait(11),
                deleteTCB(12)
            }
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The state of this TCP connection.

            The only value which may be set by a management
            station is deleteTCB(12).  Accordingly, it is
            appropriate for an agent to return a `badValue'
            response if a management station attempts to set
            this object to any other value.

            If a management station sets this object to the
            value deleteTCB(12), then this has the effect of
            deleting the TCB (as defined in RFC 793) of the
            corresponding connection on the managed node,
            resulting in immediate termination of the
            connection.

            As an implementation-specific option, a RST
```

```
                    segment may be sent from the managed node to the
                    other TCP endpoint (note however that RST segments
                    are not sent reliably)."
            ::= { tcpConnEntry 1 }

        tcpConnLocalAddress OBJECT-TYPE
            SYNTAX  IpAddress
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                    "The local IP address for this TCP connection.  In
                    the case of a connection in the listen state which
                    is willing to accept connections for any IP
                    interface associated with the node, the value
                    0.0.0.0 is used."
            ::= { tcpConnEntry 2 }

        tcpConnLocalPort OBJECT-TYPE
            SYNTAX  INTEGER (0..65535)
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                    "The local port number for this TCP connection."
            ::= { tcpConnEntry 3 }

        tcpConnRemAddress OBJECT-TYPE
            SYNTAX  IpAddress
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                    "The remote IP address for this TCP connection."
            ::= { tcpConnEntry 4 }

        tcpConnRemPort OBJECT-TYPE
            SYNTAX  INTEGER (0..65535)
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                    "The remote port number for this TCP connection."
            ::= { tcpConnEntry 5 }


        -- additional TCP objects

        tcpInErrs OBJECT-TYPE
            SYNTAX  Counter
            ACCESS  read-only
            STATUS  mandatory
```

```
    DESCRIPTION
            "The total number of segments received in error
            (e.g., bad TCP checksums)."
    ::= { tcp 14 }

tcpOutRsts OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of TCP segments sent containing the
            RST flag."
    ::= { tcp 15 }


-- the UDP group

-- Implementation of the UDP group is mandatory for all
-- systems which implement the UDP.

udpInDatagrams OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of UDP datagrams delivered to
            UDP users."
    ::= { udp 1 }

udpNoPorts OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of received UDP datagrams for
            which there was no application at the destination
            port."
    ::= { udp 2 }

udpInErrors OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of received UDP datagrams that could
            not be delivered for reasons other than the lack
            of an application at the destination port."
    ::= { udp 3 }
```

```
udpOutDatagrams OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of UDP datagrams sent from this
            entity."
    ::= { udp 4 }


-- the UDP Listener table

-- The UDP listener table contains information about this
-- entity's UDP end-points on which a local application is
-- currently accepting datagrams.

udpTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF UdpEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "A table containing UDP listener information."
    ::= { udp 5 }

udpEntry OBJECT-TYPE
    SYNTAX  UdpEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "Information about a particular current UDP
            listener."
    INDEX   { udpLocalAddress, udpLocalPort }
    ::= { udpTable 1 }

UdpEntry ::=
    SEQUENCE {
        udpLocalAddress
            IpAddress,
        udpLocalPort
            INTEGER (0..65535)
    }

udpLocalAddress OBJECT-TYPE
    SYNTAX  IpAddress
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The local IP address for this UDP listener.  In
```

```
                the case of a UDP listener which is willing to
                accept datagrams for any IP interface associated
                with the node, the value 0.0.0.0 is used."
        ::= { udpEntry 1 }

    udpLocalPort OBJECT-TYPE
        SYNTAX  INTEGER (0..65535)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The local port number for this UDP listener."
        ::= { udpEntry 2 }


    -- the EGP group

    -- Implementation of the EGP group is mandatory for all
    -- systems which implement the EGP.

    egpInMsgs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of EGP messages received without
                error."
        ::= { egp 1 }

    egpInErrors OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of EGP messages received that proved
                to be in error."
        ::= { egp 2 }

    egpOutMsgs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The total number of locally generated EGP
                messages."
        ::= { egp 3 }

    egpOutErrors OBJECT-TYPE
        SYNTAX  Counter
```

```
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of locally generated EGP messages not
            sent due to resource limitations within an EGP
            entity."
    ::= { egp 4 }


-- the EGP Neighbor table

-- The EGP neighbor table contains information about this
-- entity's EGP neighbors.

egpNeighTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF EgpNeighEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "The EGP neighbor table."
    ::= { egp 5 }

egpNeighEntry OBJECT-TYPE
    SYNTAX  EgpNeighEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "Information about this entity's relationship with
            a particular EGP neighbor."
    INDEX   { egpNeighAddr }
    ::= { egpNeighTable 1 }

EgpNeighEntry ::=
    SEQUENCE {
        egpNeighState
            INTEGER,
        egpNeighAddr
            IpAddress,
        egpNeighAs
            INTEGER,
        egpNeighInMsgs
            Counter,
        egpNeighInErrs
            Counter,
        egpNeighOutMsgs
            Counter,
        egpNeighOutErrs
            Counter,
```

```
            egpNeighInErrMsgs
                Counter,
            egpNeighOutErrMsgs
                Counter,
            egpNeighStateUps
                Counter,
            egpNeighStateDowns
                Counter,
            egpNeighIntervalHello
                INTEGER,
            egpNeighIntervalPoll
                INTEGER,
            egpNeighMode
                INTEGER,
            egpNeighEventTrigger
                INTEGER
        }

    egpNeighState OBJECT-TYPE
        SYNTAX  INTEGER {
                    idle(1),
                    acquisition(2),
                    down(3),
                    up(4),
                    cease(5)
                }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The EGP state of the local system with respect to
                this entry's EGP neighbor.  Each EGP state is
                represented by a value that is one greater than
                the numerical value associated with said state in
                RFC 904."
        ::= { egpNeighEntry 1 }

    egpNeighAddr OBJECT-TYPE
        SYNTAX  IpAddress
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The IP address of this entry's EGP neighbor."
        ::= { egpNeighEntry 2 }

    egpNeighAs OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
```

```
        DESCRIPTION
                "The autonomous system of this EGP peer.  Zero
                should be specified if the autonomous system
                number of the neighbor is not yet known."
        ::= { egpNeighEntry 3 }

    egpNeighInMsgs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of EGP messages received without error
                from this EGP peer."
        ::= { egpNeighEntry 4 }

    egpNeighInErrs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of EGP messages received from this EGP
                peer that proved to be in error (e.g., bad EGP
                checksum)."
        ::= { egpNeighEntry 5 }

    egpNeighOutMsgs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of locally generated EGP messages to
                this EGP peer."
        ::= { egpNeighEntry 6 }

    egpNeighOutErrs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of locally generated EGP messages not
                sent to this EGP peer due to resource limitations
                within an EGP entity."
        ::= { egpNeighEntry 7 }

    egpNeighInErrMsgs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
```

```
    DESCRIPTION
            "The number of EGP-defined error messages received
            from this EGP peer."
    ::= { egpNeighEntry 8 }

egpNeighOutErrMsgs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of EGP-defined error messages sent to
            this EGP peer."
    ::= { egpNeighEntry 9 }

egpNeighStateUps OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of EGP state transitions to the UP
            state with this EGP peer."
    ::= { egpNeighEntry 10 }

egpNeighStateDowns OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of EGP state transitions from the UP
            state to any other state with this EGP peer."
    ::= { egpNeighEntry 11 }

egpNeighIntervalHello OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The interval between EGP Hello command
            retransmissions (in hundredths of a second).  This
            represents the t1 timer as defined in RFC 904."
    ::= { egpNeighEntry 12 }

egpNeighIntervalPoll OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The interval between EGP poll command
```

```
                retransmissions (in hundredths of a second).  This
                represents the t3 timer as defined in RFC 904."
        ::= { egpNeighEntry 13 }

    egpNeighMode OBJECT-TYPE
        SYNTAX  INTEGER { active(1), passive(2) }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The polling mode of this EGP entity, either
                passive or active."
        ::= { egpNeighEntry 14 }

    egpNeighEventTrigger OBJECT-TYPE
        SYNTAX  INTEGER { start(1), stop(2) }
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "A control variable used to trigger operator-
                initiated Start and Stop events.  When read, this
                variable always returns the most recent value that
                egpNeighEventTrigger was set to.  If it has not
                been set since the last initialization of the
                network management subsystem on the node, it
                returns a value of `stop'.

                When set, this variable causes a Start or Stop
                event on the specified neighbor, as specified on
                pages 8-10 of RFC 904.  Briefly, a Start event
                causes an Idle peer to begin neighbor acquisition
                and a non-Idle peer to reinitiate neighbor
                acquisition.  A stop event causes a non-Idle peer
                to return to the Idle state until a Start event
                occurs, either via egpNeighEventTrigger or
                otherwise."
        ::= { egpNeighEntry 15 }


    -- additional EGP objects

    egpAs OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The autonomous system number of this EGP entity."
        ::= { egp 6 }
```

```
-- the Transmission group

-- Based on the transmission media underlying each interface
-- on a system, the corresponding portion of the Transmission
-- group is mandatory for that system.

-- When Internet-standard definitions for managing
-- transmission media are defined, the transmission group is
-- used to provide a prefix for the names of those objects.

-- Typically, such definitions reside in the experimental
-- portion of the MIB until they are "proven", then as a
-- part of the Internet standardization process, the
-- definitions are accordingly elevated and a new object
-- identifier, under the transmission group is defined. By
-- convention, the name assigned is:
--
--     type OBJECT IDENTIFIER    ::= { transmission number }
--
-- where "type" is the symbolic value used for the media in
-- the ifType column of the ifTable object, and "number" is
-- the actual integer value corresponding to the symbol.


-- the SNMP group

-- Implementation of the SNMP group is mandatory for all
-- systems which support an SNMP protocol entity.  Some of
-- the objects defined below will be zero-valued in those
-- SNMP implementations that are optimized to support only
-- those functions specific to either a management agent or
-- a management station.  In particular, it should be
-- observed that the objects below refer to an SNMP entity,
-- and there may be several SNMP entities residing on a
-- managed node (e.g., if the node is hosting acting as
-- a management station).

snmpInPkts OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of Messages delivered to the
            SNMP entity from the transport service."
    ::= { snmp 1 }

snmpOutPkts OBJECT-TYPE
    SYNTAX  Counter
```

```
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The total number of SNMP Messages which were
                passed from the SNMP protocol entity to the
                transport service."
        ::= { snmp 2 }

    snmpInBadVersions OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The total number of SNMP Messages which were
                delivered to the SNMP protocol entity and were for
                an unsupported SNMP version."
        ::= { snmp 3 }

    snmpInBadCommunityNames OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The total number of SNMP Messages delivered to
                the SNMP protocol entity which used a SNMP
                community name not known to said entity."
        ::= { snmp 4 }

    snmpInBadCommunityUses OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The total number of SNMP Messages delivered to
                the SNMP protocol entity which represented an SNMP
                operation which was not allowed by the SNMP
                community named in the Message."
        ::= { snmp 5 }

    snmpInASNParseErrs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The total number of ASN.1 or BER errors
                encountered by the SNMP protocol entity when
                decoding received SNMP Messages."
        ::= { snmp 6 }
```

```
-- { snmp 7 } is not used

snmpInTooBigs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of SNMP PDUs which were
            delivered to the SNMP protocol entity and for
            which the value of the error-status field is
            `tooBig'."
    ::= { snmp 8 }

snmpInNoSuchNames OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of SNMP PDUs which were
            delivered to the SNMP protocol entity and for
            which the value of the error-status field is
            `noSuchName'."
    ::= { snmp 9 }

snmpInBadValues OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of SNMP PDUs which were
            delivered to the SNMP protocol entity and for
            which the value of the error-status field is
            `badValue'."
    ::= { snmp 10 }

snmpInReadOnlys OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number valid SNMP PDUs which were
            delivered to the SNMP protocol entity and for
            which the value of the error-status field is
            `readOnly'.  It should be noted that it is a
            protocol error to generate an SNMP PDU which
            contains the value `readOnly' in the error-status
            field, as such this object is provided as a means
            of detecting incorrect implementations of the
```

```
                SNMP."
        ::= { snmp 11 }

    snmpInGenErrs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The total number of SNMP PDUs which were
                delivered to the SNMP protocol entity and for
                which the value of the error-status field is
                `genErr'."
        ::= { snmp 12 }

    snmpInTotalReqVars OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The total number of MIB objects which have been
                retrieved successfully by the SNMP protocol entity
                as the result of receiving valid SNMP Get-Request
                and Get-Next PDUs."
        ::= { snmp 13 }

    snmpInTotalSetVars OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The total number of MIB objects which have been
                altered successfully by the SNMP protocol entity
                as the result of receiving valid SNMP Set-Request
                PDUs."
        ::= { snmp 14 }

    snmpInGetRequests OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The total number of SNMP Get-Request PDUs which
                have been accepted and processed by the SNMP
                protocol entity."
        ::= { snmp 15 }

    snmpInGetNexts OBJECT-TYPE
        SYNTAX  Counter
```

```
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of SNMP Get-Next PDUs which have
            been accepted and processed by the SNMP protocol
            entity."
    ::= { snmp 16 }

snmpInSetRequests OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of SNMP Set-Request PDUs which
            have been accepted and processed by the SNMP
            protocol entity."
    ::= { snmp 17 }

snmpInGetResponses OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of SNMP Get-Response PDUs which
            have been accepted and processed by the SNMP
            protocol entity."
    ::= { snmp 18 }

snmpInTraps OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of SNMP Trap PDUs which have
            been accepted and processed by the SNMP protocol
            entity."
    ::= { snmp 19 }

snmpOutTooBigs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of SNMP PDUs which were
            generated by the SNMP protocol entity and for
            which the value of the error-status field is
            `tooBig.'"
    ::= { snmp 20 }
```

```
snmpOutNoSuchNames OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of SNMP PDUs which were
            generated by the SNMP protocol entity and for
            which the value of the error-status is
            'noSuchName'."
    ::= { snmp 21 }

snmpOutBadValues OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of SNMP PDUs which were
            generated by the SNMP protocol entity and for
            which the value of the error-status field is
            'badValue'."
    ::= { snmp 22 }

-- { snmp 23 } is not used

snmpOutGenErrs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of SNMP PDUs which were
            generated by the SNMP protocol entity and for
            which the value of the error-status field is
            'genErr'."
    ::= { snmp 24 }

snmpOutGetRequests OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of SNMP Get-Request PDUs which
            have been generated by the SNMP protocol entity."
    ::= { snmp 25 }

snmpOutGetNexts OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
```

```
    DESCRIPTION
            "The total number of SNMP Get-Next PDUs which have
            been generated by the SNMP protocol entity."
    ::= { snmp 26 }

snmpOutSetRequests OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of SNMP Set-Request PDUs which
            have been generated by the SNMP protocol entity."
    ::= { snmp 27 }

snmpOutGetResponses OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of SNMP Get-Response PDUs which
            have been generated by the SNMP protocol entity."
    ::= { snmp 28 }

snmpOutTraps OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of SNMP Trap PDUs which have
            been generated by the SNMP protocol entity."
    ::= { snmp 29 }

snmpEnableAuthenTraps OBJECT-TYPE
    SYNTAX  INTEGER { enabled(1), disabled(2) }
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "Indicates whether the SNMP agent process is
            permitted to generate authentication-failure
            traps.  The value of this object overrides any
            configuration information; as such, it provides a
            means whereby all authentication-failure traps may
            be disabled.

            Note that it is strongly recommended that this
            object be stored in non-volatile memory so that it
            remains constant between re-initializations of the
            network management system."
```

```
            ::= { snmp 30 }

        END
```

7. Acknowledgements

This document was produced by the SNMP Working Group:

Anne Ambler, Spider
Karl Auerbach, Sun
Fred Baker, ACC
David Bridgham, Epilogue Technology
Ken Brinkerhoff
Ron Broersma, NOSC
Brian Brown, Synoptics
Jack Brown, US Army
Theodore Brunner, Bellcore
Jeff Buffum, HP
Jeffrey Buffum, HP
John Burress, Wellfleet
Jeffrey D. Case, University of Tennessee at Knoxville
Chris Chiptasso, Spartacus
Paul Ciarfella, DEC
Bob Collet
John Cook, Chipcom
Tracy Cox, Bellcore
James R. Davin, MIT-LCS
Eric Decker, cisco
Kurt Dobbins, Cabletron
Nadya El-Afandi, Network Systems
Gary Ellis, HP
Fred Engle
Mike Erlinger
Mark S. Fedor, PSI
Richard Fox, Synoptics
Karen Frisa, CMU
Stan Froyd, ACC
Chris Gunner, DEC
Fred Harris, University of Tennessee at Knoxville
Ken Hibbard, Xylogics
Ole Jacobsen, Interop
Ken Jones
Satish Joshi, Synoptics
Frank Kastenholz, Racal-Interlan
Shimshon Kaufman, Spartacus
Ken Key, University of Tennessee at Knoxville
Jim Kinder, Fibercom
Alex Koifman, BBN

Christopher Kolb, PSI
Cheryl Krupczak, NCR
Paul Langille, DEC
Martin Lee Schoffstall, PSI
Peter Lin, Vitalink
John Lunny, TWG
Carl Malamud
Gary Malkin, FTP Software, Inc.
Randy Mayhew, University of Tennessee at Knoxville
Keith McCloghrie, Hughes LAN Systems
Donna McMaster, David Systems
Lynn Monsanto, Sun
Dave Perkins, 3COM
Jim Reinstedler, Ungerman Bass
Anil Rijsinghani, DEC
Kathy Rinehart, Arnold AFB
Kary Robertson
Marshall T. Rose, PSI (chair)
L. Michael Sabo, NCSC
Jon Saperia, DEC
Greg Satz, cisco
Martin Schoffstall, PSI
John Seligson
Steve Sherry, Xyplex
Fei Shu, NEC
Sam Sjogren, TGV
Mark Sleeper, Sparta
Lance Sprung
Mike St.Johns
Bob Stewart, Xyplex
Emil Sturniold
Kaj Tesink, Bellcore
Geoff Thompson, Synoptics
Dean Throop, Data General
Bill Townsend, Xylogics
Maurice Turcotte, Racal-Milgo
Kannan Varadhou
Sudhanshu Verma, HP
Bill Versteeg, Network Research Corporation
Warren Vik, Interactive Systems
David Waitzman, BBN
Steve Waldbusser, CMU
Dan Wintringhan
David Wood
Wengyik Yeong, PSI
Jeff Young, Cray Research

In addition, the comments of the following individuals are also acknolwedged:

Craig A. Finseth, Minnesota Supercomputer Center, Inc.
Jeffrey C. Honig, Cornell University Theory Center
Philip R. Karn, Bellcore

8. References

[1] Cerf, V., "IAB Recommendations for the Development of Internet Network Management Standards", RFC 1052, NRI, April 1988.

[2] Rose M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based internets," RFC 1065, TWG, August 1988.

[3] McCloghrie, K., and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets, RFC 1066, TWG, August 1988.

[4] Cerf, V., "Report of the Second Ad Hoc Network Management Review Group", RFC 1109, NRI, August 1989.

[5] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol (SNMP)", RFC 1098, University of Tennessee at Knoxville, NYSERNet, Inc., Rensselaer Polytechnic Institute, MIT Laboratory for Computer Science, April 1989.

[6] Postel, J., and J. Reynolds, "TELNET Protocol Specification", RFC 854, USC/Information Sciences Institute, May 1983.

[7] Satz, G., "Connectionless Network Protocol (ISO 8473) and End System to Intermediate System (ISO 9542) Management Information Base", RFC 1162, cisco Systems, Inc., June 1990.

[8] Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1), International Organization for Standardization, International Standard 8824, December 1987.

[9] Information processing systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Notation One (ASN.1), International Organization for Standardization, International Standard 8825, December 1987.

[10] Jacobson, V., "Congestion Avoidance and Control", SIGCOMM 1988, Stanford, California.

[11] Hagens, R., Hall, N., and M. Rose, "Use of the Internet as a Subnetwork for Experimentation with the OSI Network Layer", RFC 1070, U of Wiscsonsin - Madison, U of Wiscsonsin - Madison, The Wollongong Group, February 1989.

[12] Rose M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based internets", RFC 1155, Performance Systems International, Hughes LAN Systems, May 1990.

[13] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol", RFC 1157, SNMP Research, Performance Systems International, Performance Systems International, MIT Laboratory for Computer Science, May 1990.

[14] Rose, M., and K. McCloghrie, Editors, "Concise MIB Definitions", RFC 1212, Performance Systems International, Hughes LAN Systems, March 1991.

9. Security Considerations

Security issues are not discussed in this memo.

10. Authors' Addresses

Keith McCloghrie
Hughes LAN Systems
1225 Charleston Road
Mountain View, CA 94043
1225 Charleston Road
Mountain View, CA 94043

Phone: (415) 966-7934

EMail: kzm@hls.com

Marshall T. Rose
Performance Systems International
5201 Great America Parkway
Suite 3106
Santa Clara, CA 95054

Phone: +1 408 562 6222

EMail: mrose@psi.com
X.500: rose, psi, us

Network Working Group
Request for Comments: 1447

K. McCloghrie
Hughes LAN Systems
J. Galvin
Trusted Information Systems

April 1993

Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2)

Status of this Memo

This RFC specifes an IAB standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "IAB Official Protocol Standards" for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Table of Contents

1. Introduction

A network management system contains: several (potentially many) nodes, each with a processing entity, termed an agent, which has access to management instrumentation; at least one management station; and, a management protocol, used to convey management information between the agents and management stations. Operations of the protocol are carried out under an administrative framework which defines both authentication and authorization policies.

Network management stations execute management applications which monitor and control network elements. Network elements are devices such as hosts, routers, terminal servers, etc., which are monitored and controlled through access to their management information.

Management information is viewed as a collection of managed objects, residing in a virtual information store, termed the Management Information Base (MIB). Collections of related objects are defined in MIB modules. These modules are written using a subset of OSI's Abstract Syntax Notation One (ASN.1) [1], termed the Structure of Management Information (SMI) [2].

The Administrative Model for SNMPv2 document [3] defines the properties associated with SNMPv2 parties, SNMPv2 contexts, and access control policies. It is the purpose of this document, the Party MIB for SNMPv2, to define managed objects which correspond to these properties.

1.1. A Note on Terminology

For the purpose of exposition, the original Internet-standard Network Management Framework, as described in RFCs 1155, 1157, and 1212, is termed the SNMP version 1 framework (SNMPv1). The current framework is termed the SNMP version 2 framework (SNMPv2).

2. Definitions

```
SNMPv2-PARTY-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, snmpModules,
        UInteger32
        FROM SNMPv2-SMI
    TEXTUAL-CONVENTION, RowStatus, TruthValue
        FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP
        FROM SNMPv2-CONF;


partyMIB MODULE-IDENTITY
    LAST-UPDATED "9304010000Z"
    ORGANIZATION "IETF SNMP Security Working Group"
    CONTACT-INFO
            "       Keith McCloghrie

             Postal: Hughes LAN Systems
                     1225 Charleston Road
                     Mountain View, CA  94043
                     US

                Tel: +1 415 966 7934
                Fax: +1 415 960 3738

             E-mail: kzm@hls.com"
    DESCRIPTION
            "The MIB module describing SNMPv2 parties."
    ::= { snmpModules 3 }
```

```
-- textual conventions

Party ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
            "Denotes a SNMPv2 party identifier.

            Note that agents may impose implementation
            limitations on the length of OIDs used to identify
            Parties. As such, management stations creating
            new parties should be aware that using an
            excessively long OID may result in the agent
            refusing to perform the set operation and instead
            returning the appropriate error response, e.g.,
            noCreation."
    SYNTAX       OBJECT IDENTIFIER


TAddress ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
            "Denotes a transport service address.

            For snmpUDPDomain, a TAddress is 6 octets long,
            the initial 4 octets containing the IP-address in
            network-byte order and the last 2 containing the
            UDP port in network-byte order.  Consult [5] for
            further information on snmpUDPDomain."
    SYNTAX       OCTET STRING
```

```
Clock ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
            "A party's authentication clock - a non-negative
            integer which is incremented as specified/allowed
            by the party's Authentication Protocol.

            For noAuth, a party's authentication clock is
            unused and its value is undefined.

            For v2md5AuthProtocol, a party's authentication
            clock is a relative clock with 1-second
            granularity."
    SYNTAX       UInteger32


Context ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
            "Denotes a SNMPv2 context identifier.

            Note that agents may impose implementation
            limitations on the length of OIDs used to identify
            Contexts. As such, management stations creating new
            contexts should be aware that using an excessively
            long OID may result in the agent refusing to
            perform the set operation and instead returning
            the appropriate error response, e.g., noCreation."
    SYNTAX       OBJECT IDENTIFIER
```

```
StorageType ::= TEXTUAL-CONVENTION
    STATUS        current
    DESCRIPTION
            "Describes the memory realization of a conceptual
            row.  A row which is volatile(2) is lost upon
            reboot.  A row which is nonVolatile(3) is backed
            up by stable storage.  A row which is permanent(4)
            cannot be changed nor deleted."
    SYNTAX        INTEGER {
                      other(1),        -- eh?
                      volatile(2),     -- e.g., in RAM
                      nonVolatile(3),  -- e.g., in NVRAM
                      permanent(4)     -- e.g., in ROM
                  }
```



```
-- administrative assignments
```

```
partyAdmin     OBJECT IDENTIFIER ::= { partyMIB 1 }


-- definitions of security protocols

partyProtocols OBJECT IDENTIFIER ::= { partyAdmin 1 }

-- the protocol without authentication
noAuth         OBJECT IDENTIFIER ::= { partyProtocols 1 }

-- the protocol without privacy
noPriv         OBJECT IDENTIFIER ::= { partyProtocols 2 }

-- the DES Privacy Protocol [4]
desPrivProtocol
               OBJECT IDENTIFIER ::= { partyProtocols 3 }

-- the MD5 Authentication Protocol [4]
v2md5AuthProtocol
               OBJECT IDENTIFIER ::= { partyProtocols 4 }


-- definitions of temporal domains

temporalDomains
               OBJECT IDENTIFIER ::= { partyAdmin 2 }

-- this temporal domain refers to management information
-- at the current time
currentTime    OBJECT IDENTIFIER ::= { temporalDomains 1 }

-- this temporal domain refers to management information
-- upon the next re-initialization of the managed device
restartTime    OBJECT IDENTIFIER ::= { temporalDomains 2 }

-- the temporal domain { cacheTime N } refers to management
-- information that is cached and guaranteed to be at most
-- N seconds old
cacheTime      OBJECT IDENTIFIER ::= { temporalDomains 3 }
```



```
-- Definition of Initial Party and Context Identifiers
```

```
-- When devices are installed, they need to be configured
-- with an initial set of SNMPv2 parties and contexts.  The
-- configuration of SNMPv2 parties and contexts requires (among
-- other things) the assignment of several OBJECT IDENTIFIERs.
-- Any local network administration can obtain the delegated
-- authority necessary to assign its own OBJECT IDENTIFIERs.
-- However, to provide for those administrations who have not
-- obtained the necessary authority, this document allocates a
-- branch of the naming tree for use with the following
-- conventions.

initialPartyId OBJECT IDENTIFIER ::= { partyAdmin 3 }

initialContextId
               OBJECT IDENTIFIER ::= { partyAdmin 4 }

-- Note these are identified as "initial" party and context
-- identifiers since these allow secure SNMPv2 communication
-- to proceed, thereby allowing further SNMPv2 parties to be
-- configured through use of the SNMPv2 itself.

-- The following definitions identify a party identifier, and
-- specify the initial values of various object instances
-- indexed by that identifier.  In addition, the SNMPv2
-- context, access control policy, and MIB view information
-- assigned, by convention, are identified.
```



```
-- Party Identifiers for use as initial SNMPv2 parties
--       at IP address  a.b.c.d
```

```
-- Note that for all OBJECT IDENTIFIERs assigned under
-- initialPartyId, the four sub-identifiers immediately
-- following initialPartyId represent the four octets of
-- an IP address.  Initial party identifiers for other address
-- families are assigned under a different OBJECT IDENTIFIER,
-- as defined elsewhere.

-- Devices which support SNMPv2 as entities acting in an
-- agent role, and accessed via the snmpUDPDomain transport
-- domain, are required to be configured with the appropriate
-- set of the following as implicit assignments as and when
-- they are configured with an IP address.  The appropriate
-- set is all those applicable to the authentication and
-- privacy protocols supported by the device.
```



```
--      a noAuth/noPriv party which executes at the agent
-- partyIdentity           = { initialPartyId a b c d 1 }
-- partyIndex              = 1
-- partyTDomain            = snmpUDPDomain
```

```
-- partyTAddress            = a.b.c.d, 161
-- partyLocal               = true (in agent's database)
-- partyAuthProtocol        = noAuth
-- partyAuthClock           = 0
-- partyAuthPrivate         = ''H     (the empty string)
-- partyAuthPublic          = ''H     (the empty string)
-- partyAuthLifetime        = 0
-- partyPrivProtocol        = noPriv
-- partyPrivPrivate         = ''H     (the empty string)
-- partyPrivPublic          = ''H     (the empty string)

--      a noAuth/noPriv party which executes at a manager
-- partyIdentity            = { initialPartyId a b c d 2 }
-- partyIndex               = 2
-- partyTDomain             = snmpUDPDomain
-- partyTAddress            = assigned by local administration
-- partyLocal               = false (in agent's database)
-- partyAuthProtocol        = noAuth
-- partyAuthClock           = 0
-- partyAuthPrivate         = ''H     (the empty string)
-- partyAuthPublic          = ''H     (the empty string)
-- partyAuthLifetime        = 0
-- partyPrivProtocol        = noPriv
-- partyPrivPrivate         = ''H     (the empty string)
-- partyPrivPublic          = ''H     (the empty string)
```



```
--      a md5Auth/noPriv party which executes at the agent
-- partyIdentity            = { initialPartyId a b c d 3 }
-- partyIndex               = 3
-- partyTDomain             = snmpUDPDomain
-- partyTAddress            = a.b.c.d, 161
```

```
-- partyLocal                 = true (in agent's database)
-- partyAuthProtocol          = v2md5AuthProtocol
-- partyAuthClock             = 0
-- partyAuthPrivate           = assigned by local administration
-- partyAuthPublic            = ''H    (the empty string)
-- partyAuthLifetime          = 300
-- partyPrivProtocol          = noPriv
-- partyPrivPrivate           = ''H    (the empty string)
-- partyPrivPublic            = ''H    (the empty string)

--      a md5Auth/noPriv party which executes at a manager
-- partyIdentity              = { initialPartyId a b c d 4 }
-- partyIndex                 = 4
-- partyTDomain               = snmpUDPDomain
-- partyTAddress              = assigned by local administration
-- partyLocal                 = false (in agent's database)
-- partyAuthProtocol          = v2md5AuthProtocol
-- partyAuthClock             = 0
-- partyAuthPrivate           = assigned by local administration
-- partyAuthPublic            = ''H    (the empty string)
-- partyAuthLifetime          = 300
-- partyPrivProtocol          = noPriv
-- partyPrivPrivate           = ''H    (the empty string)
-- partyPrivPublic            = ''H    (the empty string)
```



```
--      a md5Auth/desPriv party which executes at the agent
-- partyIdentity              = { initialPartyId a b c d 5 }
-- partyIndex                 = 5
-- partyTDomain               = snmpUDPDomain
-- partyTAddress              = a.b.c.d, 161
-- partyLocal                 = true (in agent's database)
```

```
-- partyAuthProtocol        = v2md5AuthProtocol
-- partyAuthClock           = 0
-- partyAuthPrivate         = assigned by local administration
-- partyAuthPublic          = ''H    (the empty string)
-- partyAuthLifetime        = 300
-- partyPrivProtocol        = desPrivProtocol
-- partyPrivPrivate         = assigned by local administration
-- partyPrivPublic          = ''H    (the empty string)

--      a md5Auth/desPriv party which executes at a manager
-- partyIdentity            = { initialPartyId a b c d 6 }
-- partyIndex               = 6
-- partyTDomain             = snmpUDPDomain
-- partyTAddress            = assigned by local administration
-- partyLocal               = false (in agent's database)
-- partyAuthProtocol        = v2md5AuthProtocol
-- partyAuthClock           = 0
-- partyAuthPrivate         = assigned by local administration
-- partyAuthPublic          = ''H    (the empty string)
-- partyAuthLifetime        = 300
-- partyPrivProtocol        = desPrivProtocol
-- partyPrivPrivate         = assigned by local administration
-- partyPrivPublic          = ''H    (the empty string)
```



```
-- the initial SNMPv2 contexts assigned, by convention, are:

-- contextIdentity          = { initialContextId a b c d 1 }
-- contextIndex             = 1
-- contextLocal             = true (in agent's database)
-- contextViewIndex         = 1
-- contextLocalEntity       = ''H    (the empty string)
```

```
-- contextLocalTime          = currentTime
-- contextProxyDstParty      = { 0 0 }
-- contextProxySrcParty      = { 0 0 }
-- contextProxyContext       = { 0 0 }

-- contextIdentity           = { initialContextId a b c d 2 }
-- contextIndex              = 2
-- contextLocal              = true (in agent's database)
-- contextViewIndex          = 2
-- contextLocalEntity        = ''H    (the empty string)
-- contextLocalTime          = currentTime
-- contextProxyDstParty      = { 0 0 }
-- contextProxySrcParty      = { 0 0 }
-- contextProxyContext       = { 0 0 }
```



```
-- The initial access control policy assigned, by
-- convention, is:

-- aclTarget              =   1
-- aclSubject             =   2
-- aclResources           =   1
-- aclPrivileges          =  35 (Get, Get-Next & Get-Bulk)
```

```
-- aclTarget            =   2
-- aclSubject           =   1
-- aclResources         =   1
-- aclPrivileges        = 132  (Response & SNMPv2-Trap)

-- aclTarget            =   3
-- aclSubject           =   4
-- aclResources         =   2
-- aclPrivileges        =  43  (Get, Get-Next, Set & Get-Bulk)

-- aclTarget            =   4
-- aclSubject           =   3
-- aclResources         =   2
-- aclPrivileges        =   4  (Response)

-- aclTarget            =   5
-- aclSubject           =   6
-- aclResources         =   2
-- aclPrivileges        =  43  (Get, Get-Next, Set & Get-Bulk)

-- aclTarget            =   6
-- aclSubject           =   5
-- aclResources         =   2
-- aclPrivileges        =   4  (Response)


-- Note that the initial context and access control
-- information assigned above, by default, to the
-- md5Auth/desPriv parties are identical to those assigned to
-- the md5Auth/noPriv parties.  However, each administration
-- may choose to have different authorization policies,
-- depending on whether privacy is used.
```



```
-- The initial MIB views assigned, by convention, are:

-- viewIndex              = 1
-- viewSubtree            = system
-- viewMask               = ''H
-- viewType               = included

-- viewIndex              = 1
-- viewSubtree            = snmpStats
```

```
-- viewMask                  = ''H
-- viewType                  = included

-- viewIndex                 = 1
-- viewSubtree               = snmpParties
-- viewMask                  = ''H
-- viewType                  = included

-- viewIndex                 = 2
-- viewSubtree               = internet
-- viewMask                  = ''H
-- viewType                  = included


-- Note that full access to the partyTable, contextTable,
-- aclTable, and viewTable gives a manager the ability to
-- configure any parties with any/all capabilities (the
-- equivalent of "root" access).  A lesser manager can be
-- given access only to the partyTable so that it can
-- maintain its own parties, but not increase/decrease
-- their capabilities.  Such a lesser manager can also
-- create new parties but they are of no use to it.
```



```
-- object assignments

partyMIBObjects
               OBJECT IDENTIFIER ::= { partyMIB 2 }


-- the SNMPv2 party database group

snmpParties    OBJECT IDENTIFIER ::= { partyMIBObjects 1 }
```

```
partyTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF PartyEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "The SNMPv2 Party database."
    ::= { snmpParties 1 }

partyEntry OBJECT-TYPE
    SYNTAX      PartyEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "Locally held information about a particular
            SNMPv2 party."
    INDEX      { IMPLIED partyIdentity }
    ::= { partyTable 1 }
```



```
PartyEntry ::=
    SEQUENCE {
        partyIdentity         Party,
        partyIndex            INTEGER,
        partyTDomain          OBJECT IDENTIFIER,
        partyTAddress         TAddress,
        partyMaxMessageSize   INTEGER,
        partyLocal            TruthValue,
        partyAuthProtocol     OBJECT IDENTIFIER,
        partyAuthClock        Clock,
        partyAuthPrivate      OCTET STRING,
```

```
        partyAuthPublic       OCTET STRING,
        partyAuthLifetime     INTEGER,
        partyPrivProtocol     OBJECT IDENTIFIER,
        partyPrivPrivate      OCTET STRING,
        partyPrivPublic       OCTET STRING,
        partyCloneFrom        Party,
        partyStorageType      StorageType,
        partyStatus           RowStatus
    }

partyIdentity OBJECT-TYPE
    SYNTAX      Party
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "A party identifier uniquely identifying a
            particular SNMPv2 party."
    ::= { partyEntry 1 }

partyIndex OBJECT-TYPE
    SYNTAX      INTEGER (1..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "A unique value for each SNMPv2 party.  The value
            for each SNMPv2 party must remain constant at
            least from one re-initialization of the entity's
            network management system to the next re-
            initialization."
    ::= { partyEntry 2 }
```



```
partyTDomain OBJECT-TYPE
    SYNTAX      OBJECT IDENTIFIER
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "Indicates the kind of transport service by which
            the party receives network management traffic."
    DEFVAL      { snmpUDPDomain }
    ::= { partyEntry 3 }

partyTAddress OBJECT-TYPE
    SYNTAX      TAddress
```

```
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The transport service address by which the party
            receives network management traffic, formatted
            according to the corresponding value of
            partyTDomain.  For snmpUDPDomain, partyTAddress is
            formatted as a 4-octet IP Address concatenated
            with a 2-octet UDP port number."
    DEFVAL      { '000000000000'H }
    ::= { partyEntry 4 }

partyMaxMessageSize OBJECT-TYPE
    SYNTAX      INTEGER (484..65507)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The maximum length in octets of a SNMPv2 message
            which this party will accept.  For parties which
            execute at an agent, the agent initializes this
            object to the maximum length supported by the
            agent, and does not let the object be set to any
            larger value.  For parties which do not execute at
            the agent, the agent must allow the manager to set
            this object to any legal value, even if it is
            larger than the agent can generate."
    DEFVAL      { 484 }
    ::= { partyEntry 5 }
```



```
partyLocal OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "An indication of whether this party executes at
            this SNMPv2 entity.  If this object has a value of
            true(1), then the SNMPv2 entity will listen for
            SNMPv2 messages on the partyTAddress associated
            with this party.  If this object has the value
            false(2), then the SNMPv2 entity will not listen
            for SNMPv2 messages on the partyTAddress
            associated with this party."
```

```
    DEFVAL      { false }
    ::= { partyEntry 6 }

partyAuthProtocol OBJECT-TYPE
    SYNTAX      OBJECT IDENTIFIER
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The authentication protocol by which all messages
            generated by the party are authenticated as to
            origin and integrity.  The value noAuth signifies
            that messages generated by the party are not
            authenticated.

            Once an instance of this object is created, its
            value can not be changed."
    DEFVAL      { v2md5AuthProtocol }
    ::= { partyEntry 7 }
```



```
partyAuthClock OBJECT-TYPE
    SYNTAX      Clock
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The authentication clock which represents the
            local notion of the current time specific to the
            party.  This value must not be decremented unless
            the party's private authentication key is changed
            simultaneously."
    DEFVAL      { 0 }
    ::= { partyEntry 8 }
```

```
partyAuthPrivate OBJECT-TYPE
    SYNTAX      OCTET STRING
                -- for v2md5AuthProtocol: (SIZE (16))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "An encoding of the party's private authentication
            key which may be needed to support the
            authentication protocol.  Although the value of
            this variable may be altered by a management
            operation (e.g., a SNMPv2 Set-Request), its value
            can never be retrieved by a management operation:
            when read, the value of this variable is the zero
            length OCTET STRING.
```

```
               The private authentication key is NOT directly
               represented by the value of this variable, but
               rather it is represented according to an encoding.
               This encoding is the bitwise exclusive-OR of the
               old key with the new key, i.e., of the old private
               authentication key (prior to the alteration) with
               the new private authentication key (after the
               alteration).  Thus, when processing a received
               protocol Set operation, the new private
               authentication key is obtained from the value of
               this variable as the result of a bitwise
               exclusive-OR of the variable's value and the old
               private authentication key.  In calculating the
               exclusive-OR, if the old key is shorter than the
               new key, zero-valued padding is appended to the
               old key.  If no value for the old key exists, a
               zero-length OCTET STRING is used in the
               calculation."
       DEFVAL      { ''H }     -- the empty string
       ::= { partyEntry 9 }
```



```
   partyAuthPublic OBJECT-TYPE
       SYNTAX      OCTET STRING
                   -- for v2md5AuthProtocol: (SIZE (0..16))
       MAX-ACCESS  read-create
       STATUS      current
       DESCRIPTION
               "A publically-readable value for the party.

               Depending on the party's authentication protocol,
               this value may be needed to support the party's
               authentication protocol.  Alternatively, it may be
               used by a manager during the procedure for
               altering secret information about a party.  (For
               example, by altering the value of an instance of
               this object in the same SNMPv2 Set-Request used to
               update an instance of partyAuthPrivate, a
```

```
          subsequent Get-Request can determine if the Set-
          Request was successful in the event that no
          response to the Set-Request is received, see [4].)

          The length of the value is dependent on the
          party's authentication protocol.  If not used by
          the authentication protocol, it is recommended
          that agents support values of any length up to and
          including the length of the corresponding
          partyAuthPrivate object."
    DEFVAL      { ''H }      -- the empty string
    ::= { partyEntry 10 }
```



```
partyAuthLifetime OBJECT-TYPE
    SYNTAX      INTEGER (0..2147483647)
    UNITS       "seconds"
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The lifetime (in units of seconds) which
            represents an administrative upper bound on
            acceptable delivery delay for protocol messages
            generated by the party.

            Once an instance of this object is created, its
            value can not be changed."
    DEFVAL      { 300 }
    ::= { partyEntry 11 }

partyPrivProtocol OBJECT-TYPE
```

```
    SYNTAX      OBJECT IDENTIFIER
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The privacy protocol by which all protocol
            messages received by the party are protected from
            disclosure.  The value noPriv signifies that
            messages received by the party are not protected.

            Once an instance of this object is created, its
            value can not be changed."
    DEFVAL      { noPriv }
    ::= { partyEntry 12 }
```



```
partyPrivPrivate OBJECT-TYPE
    SYNTAX      OCTET STRING
                -- for desPrivProtocol: (SIZE (16))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "An encoding of the party's private encryption key
            which may be needed to support the privacy
            protocol.  Although the value of this variable may
            be altered by a management operation (e.g., a
            SNMPv2 Set-Request), its value can never be
            retrieved by a management operation: when read,
            the value of this variable is the zero length
            OCTET STRING.

            The private encryption key is NOT directly
            represented by the value of this variable, but
            rather it is represented according to an encoding.
```

```
                This encoding is the bitwise exclusive-OR of the
                old key with the new key, i.e., of the old private
                encryption key (prior to the alteration) with the
                new private encryption key (after the alteration).
                Thus, when processing a received protocol Set
                operation, the new private encryption key is
                obtained from the value of this variable as the
                result of a bitwise exclusive-OR of the variable's
                value and the old private encryption key.  In
                calculating the exclusive-OR, if the old key is
                shorter than the new key, zero-valued padding is
                appended to the old key.  If no value for the old
                key exists, a zero-length OCTET STRING is used in
                the calculation."
        DEFVAL      { ''H }      -- the empty string
        ::= { partyEntry 13 }
```



```
partyPrivPublic OBJECT-TYPE
    SYNTAX      OCTET STRING
                -- for desPrivProtocol: (SIZE (0..16))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "A publically-readable value for the party.

            Depending on the party's privacy protocol, this
            value may be needed to support the party's privacy
            protocol.  Alternatively, it may be used by a
            manager as a part of its procedure for altering
            secret information about a party.  (For example,
            by altering the value of an instance of this
            object in the same SNMPv2 Set-Request used to
            update an instance of partyPrivPrivate, a
            subsequent Get-Request can determine if the Set-
            Request was successful in the event that no
            response to the Set-Request is received, see [4].)
```

```
            The length of the value is dependent on the
            party's privacy protocol.  If not used by the
            privacy protocol, it is recommended that agents
            support values of any length up to and including
            the length of the corresponding partyPrivPrivate
            object."
    DEFVAL      { ''H }      -- the empty string
    ::= { partyEntry 14 }
```



```
partyCloneFrom OBJECT-TYPE
    SYNTAX      Party
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The identity of a party to clone authentication
            and privacy parameters from.  When read, the value
            { 0 0 } is returned.

            This value must be written exactly once, when the
            associated instance of partyStatus either does not
            exist or has the value `notReady'.  When written,
            the value identifies a party, the cloning party,
            whose status column has the value `active'.  The
            cloning party is used in two ways.

            One, if instances of the following objects do not
            exist for the party being created, then they are
            created with values identical to those of the
            corresponding objects for the cloning party:
```

```
                  partyAuthProtocol
                  partyAuthPublic
                  partyAuthLifetime
                  partyPrivProtocol
                  partyPrivPublic

             Two, instances of the following objects are
             updated using the corresponding values of the
             cloning party:

                  partyAuthPrivate
                  partyPrivPrivate

             (e.g., the value of the cloning party's instance
             of the partyAuthPrivate object is XOR'd with the
             value of the partyAuthPrivate instances of the
             party being created.)"
     ::= { partyEntry 15 }
```



```
partyStorageType OBJECT-TYPE
    SYNTAX      StorageType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The storage type for this conceptual row in the
            partyTable."
    DEFVAL      { nonVolatile }
    ::= { partyEntry 16 }

partyStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The status of this conceptual row in the
            partyTable.

            A party is not qualified for activation until
            instances of all columns of its partyEntry row
            have an appropriate value.  In particular:
```

```
                A value must be written to the Party's
                partyCloneFrom object.

                If the Party's partyAuthProtocol object has the
                value md5AuthProtocol, then the corresponding
                instance of partyAuthPrivate must contain a
                secret of the appropriate length.  Further, at
                least one management protocol set operation
                updating the value of the party's
                partyAuthPrivate object must be successfully
                processed, before the partyAuthPrivate column is
                considered appropriately configured.

                If the Party's partyPrivProtocol object has the
                value desPrivProtocol, then the corresponding
                instance of partyPrivPrivate must contain a
                secret of the appropriate length.  Further, at
                least one management protocol set operation
                updating the value of the party's
                partyPrivPrivate object must be successfully
                processed, before the partyPrivPrivate column is
                considered appropriately configured.
```



```
             Until instances of all corresponding columns are
             appropriately configured, the value of the
             corresponding instance of the partyStatus column is
             `notReady'."
    ::= { partyEntry 17 }
```

```
-- the SNMPv2 contexts database group

snmpContexts   OBJECT IDENTIFIER ::= { partyMIBObjects 2 }


contextTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF ContextEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "The SNMPv2 Context database."
    ::= { snmpContexts 1 }

contextEntry OBJECT-TYPE
    SYNTAX      ContextEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "Locally held information about a particular
            SNMPv2 context."
    INDEX      { IMPLIED contextIdentity }
    ::= { contextTable 1 }
```

```
ContextEntry ::=
    SEQUENCE {
        contextIdentity          Context,
        contextIndex             INTEGER,
        contextLocal             TruthValue,
        contextViewIndex         INTEGER,
        contextLocalEntity       OCTET STRING,
        contextLocalTime         OBJECT IDENTIFIER,
        contextProxyDstParty     Party,
        contextProxySrcParty     Party,
        contextProxyContext      OBJECT IDENTIFIER,
        contextStorageType       StorageType,
        contextStatus            RowStatus
    }
```



```
contextIdentity OBJECT-TYPE
    SYNTAX      Context
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "A context identifier uniquely identifying a
            particular SNMPv2 context."
    ::= { contextEntry 1 }

contextIndex OBJECT-TYPE
    SYNTAX      INTEGER (1..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "A unique value for each SNMPv2 context.  The
            value for each SNMPv2 context must remain constant
            at least from one re-initialization of the
            entity's network management system to the next
            re-initialization."
    ::= { contextEntry 2 }

contextLocal OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-create
```

```
    STATUS      current
    DESCRIPTION
            "An indication of whether this context is realized
            by this SNMPv2 entity."
    DEFVAL      { true }
    ::= { contextEntry 3 }
```



```
contextViewIndex OBJECT-TYPE
    SYNTAX      INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If the value of an instance of this object is
            zero, then this corresponding conceptual row in
            the contextTable refers to a SNMPv2 context which
            identifies a proxy relationship; the values of the
            corresponding instances of the
            contextProxyDstParty, contextProxySrcParty, and
            contextProxyContext objects provide further
            information on the proxy relationship.

            Otherwise, if the value of an instance of this
            object is greater than zero, then this
            corresponding conceptual row in the contextTable
            refers to a SNMPv2 context which identifies a MIB
            view of a locally accessible entity; the value of
            the instance identifies the particular MIB view
            which has the same value of viewIndex; and the
            value of the corresponding instances of the
            contextLocalEntity and contextLocalTime objects
            provide further information on the local entity
            and its temporal domain."
```

```
    ::= { contextEntry 4 }
```



```
contextLocalEntity OBJECT-TYPE
    SYNTAX      OCTET STRING
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If the value of the corresponding instance of the
            contextViewIndex is greater than zero, then the
            value of an instance of this object identifies the
            local entity whose management information is in
            the SNMPv2 context's MIB view.  The empty string
            indicates that the MIB view contains the SNMPv2
            entity's own local management information;
            otherwise, a non-empty string indicates that the
            MIB view contains management information of some
            other local entity, e.g., 'Repeater1'."
    DEFVAL      { ''H }     -- the empty string
    ::= { contextEntry 5 }

contextLocalTime OBJECT-TYPE
    SYNTAX      OBJECT IDENTIFIER
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If the value of the corresponding instance of the
            contextViewIndex is greater than zero, then the
            value of an instance of this object identifies the
```

```
            temporal context of the management information in
            the MIB view."
    DEFVAL      { currentTime }
    ::= { contextEntry 6 }
```



```
contextProxyDstParty OBJECT-TYPE
    SYNTAX      Party
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If the value of the corresponding instance of the
            contextViewIndex is equal to zero, then the value
            of an instance of this object identifies a SNMPv2
            party which is the proxy destination of a proxy
            relationship.

            If the value of the corresponding instance of the
            contextViewIndex is greater than zero, then the
            value of an instance of this object is { 0 0 }."
    ::= { contextEntry 7 }

contextProxySrcParty OBJECT-TYPE
    SYNTAX      Party
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If the value of the corresponding instance of the
            contextViewIndex is equal to zero, then the value
            of an instance of this object identifies a SNMPv2
            party which is the proxy source of a proxy
            relationship.
```

```
            Interpretation of an instance of this object
            depends upon the value of the transport domain
            associated with the SNMPv2 party used as the proxy
            destination in this proxy relationship.

            If the value of the corresponding instance of the
            contextViewIndex is greater than zero, then the
            value of an instance of this object is { 0 0 }."
    ::= { contextEntry 8 }
```



```
contextProxyContext OBJECT-TYPE
    SYNTAX      OBJECT IDENTIFIER
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If the value of the corresponding instance of the
            contextViewIndex is equal to zero, then the value
            of an instance of this object identifies the
            context of a proxy relationship.

            Interpretation of an instance of this object
            depends upon the value of the transport domain
            associated with the SNMPv2 party used as the proxy
            destination in this proxy relationship.

            If the value of the corresponding instance of the
            contextViewIndex is greater than zero, then the
            value of an instance of this object is { 0 0 }."
    ::= { contextEntry 9 }

contextStorageType OBJECT-TYPE
    SYNTAX      StorageType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The storage type for this conceptual row in the
            contextTable."
    DEFVAL      { nonVolatile }
```

```
    ::= { contextEntry 10 }
```



```
contextStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The status of this conceptual row in the
            contextTable.

            A context is not qualified for activation until
            instances of all corresponding columns have the
            appropriate value.  In  particular, if the
            context's contextViewIndex is greater than zero,
            then the viewStatus column of the associated
            conceptual row(s) in the viewTable must have the
            value `active'.  Until instances of all
            corresponding columns are appropriately
            configured, the value of the corresponding
            instance of the contextStatus column is
            `notReady'."
    ::= { contextEntry 11 }
```

```
-- the SNMPv2 access privileges database group

snmpAccess      OBJECT IDENTIFIER ::= { partyMIBObjects 3 }


aclTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF AclEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "The access privileges database."
    ::= { snmpAccess 1 }

aclEntry OBJECT-TYPE
    SYNTAX      AclEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "The access privileges for a particular subject
            SNMPv2 party when asking a particular target
            SNMPv2 party to access a particular SNMPv2
            context."
    INDEX      { aclTarget, aclSubject, aclResources }
    ::= { aclTable 1 }

AclEntry ::=
    SEQUENCE {
        aclTarget         INTEGER,
        aclSubject        INTEGER,
        aclResources      INTEGER,
```

```
        aclPrivileges     INTEGER,
        aclStorageType    StorageType,
        aclStatus         RowStatus
    }
```



```
aclTarget OBJECT-TYPE
    SYNTAX       INTEGER (1..65535)
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
            "The value of an instance of this object
            identifies a SNMPv2 party which is the target of
            an access control policy, and has the same value
            as the instance of the partyIndex object for that
            party."
    ::= { aclEntry 1 }

aclSubject OBJECT-TYPE
    SYNTAX       INTEGER (1..65535)
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
            "The value of an instance of this object
            identifies a SNMPv2 party which is the subject of
            an access control policy, and has the same value
            as the instance of the partyIndex object for that
            SNMPv2 party."
    ::= { aclEntry 2 }

aclResources OBJECT-TYPE
    SYNTAX       INTEGER (1..65535)
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
            "The value of an instance of this object
            identifies a SNMPv2 context in an access control
```

```
            policy, and has the same value as the instance of
            the contextIndex object for that SNMPv2 context."
    ::= { aclEntry 3 }
```



```
aclPrivileges OBJECT-TYPE
    SYNTAX      INTEGER (0..255)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The access privileges which govern what
            management operations a particular target party
            may perform with respect to a particular SNMPv2
            context when requested by a particular subject
            party.  These privileges are specified as a sum of
            values, where each value specifies a SNMPv2 PDU
            type by which the subject party may request a
            permitted operation.  The value for a particular
            PDU type is computed as 2 raised to the value of
            the ASN.1 context-specific tag for the appropriate
            SNMPv2 PDU type.  The values (for the tags defined
            in [5]) are defined in [3] as:

             Get         :   1
             GetNext     :   2
             Response    :   4
             Set         :   8
             unused      :  16
             GetBulk     :  32
             Inform      :  64
             SNMPv2-Trap : 128

            The null set is represented by the value zero."
    DEFVAL      { 35 }      -- Get, Get-Next & Get-Bulk
    ::= { aclEntry 4 }

aclStorageType OBJECT-TYPE
```

```
    SYNTAX      StorageType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The storage type for this conceptual row in the
            aclTable."
    DEFVAL      { nonVolatile }
    ::= { aclEntry 5 }
```



```
aclStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The status of this conceptual row in the
            aclTable."
    ::= { aclEntry 6 }
```

```
-- the MIB view database group

snmpViews      OBJECT IDENTIFIER ::= { partyMIBObjects 4 }


viewTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF ViewEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "Locally held information about the MIB views
            known to this SNMPv2 entity.

            Each SNMPv2 context which is locally accessible
            has a single MIB view which is defined by two
            collections of view subtrees: the included view
            subtrees, and the excluded view subtrees.  Every
            such subtree, both included and excluded, is
            defined in this table.

            To determine if a particular object instance is in
            a particular MIB view, compare the object
            instance's OBJECT IDENTIFIER with each of the MIB
            view's entries in this table.  If none match, then
            the object instance is not in the MIB view.  If
            one or more match, then the object instance is
            included in, or excluded from, the MIB view
            according to the value of viewType in the entry
            whose value of viewSubtree has the most sub-
            identifiers.  If multiple entries match and have
            the same number of sub-identifiers, then the
            lexicographically greatest instance of viewType
            determines the inclusion or exclusion.
```

```
            An object instance's OBJECT IDENTIFIER X matches
            an entry in this table when the number of sub-
            identifiers in X is at least as many as in the
            value of viewSubtree for the entry, and each sub-
            identifier in the value of viewSubtree matches its
            corresponding sub-identifier in X.  Two sub-
            identifiers match either if the corresponding bit
            of viewMask is zero (the 'wild card' value), or if
            they are equal.

            Due to this 'wild card' capability, we introduce
```



```
            the term, a 'family' of view subtrees, to refer to
            the set of subtrees defined by a particular
            combination of values of viewSubtree and viewMask.
            In the case where no 'wild card' is defined in
            viewMask, the family of view subtrees reduces to a
            single view subtree."
    ::= { snmpViews 1 }

viewEntry OBJECT-TYPE
    SYNTAX      ViewEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "Information on a particular family of view
            subtrees included in or excluded from a particular
            SNMPv2 context's MIB view.

            Implementations must not restrict the number of
            families of view subtrees for a given MIB view,
            except as dictated by resource constraints on the
            overall number of entries in the viewTable."
    INDEX      { viewIndex, IMPLIED viewSubtree }
    ::= { viewTable 1 }

ViewEntry ::=
    SEQUENCE {
        viewIndex         INTEGER,
        viewSubtree       OBJECT IDENTIFIER,
        viewMask          OCTET STRING,
        viewType          INTEGER,
        viewStorageType   StorageType,
        viewStatus        RowStatus
    }
```

```
viewIndex OBJECT-TYPE
    SYNTAX      INTEGER (1..65535)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "A unique value for each MIB view.  The value for
            each MIB view must remain constant at least from
            one re-initialization of the entity's network
            management system to the next re-initialization."
    ::= { viewEntry 1 }

viewSubtree OBJECT-TYPE
    SYNTAX      OBJECT IDENTIFIER
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "A MIB subtree."
    ::= { viewEntry 2 }

viewMask OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE (0..16))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The bit mask which, in combination with the
            corresponding instance of viewSubtree, defines a
            family of view subtrees.

            Each bit of this bit mask corresponds to a sub-
            identifier of viewSubtree, with the most
            significant bit of the i-th octet of this octet
            string value (extended if necessary, see below)
            corresponding to the (8*i - 7)-th sub-identifier,
            and the least significant bit of the i-th octet of
            this octet string corresponding to the (8*i)-th
            sub-identifier, where i is in the range 1 through
```

16.

Each bit of this bit mask specifies whether or not the corresponding sub-identifiers must match when determining if an OBJECT IDENTIFIER is in this family of view subtrees; a '1' indicates that an exact match must occur; a '0' indicates 'wild card', i.e., any sub-identifier value matches.

Thus, the OBJECT IDENTIFIER X of an object instance is contained in a family of view subtrees if the following criteria are met:

for each sub-identifier of the value of viewSubtree, either:

the i-th bit of viewMask is 0, or

the i-th sub-identifier of X is equal to the i-th sub-identifier of the value of viewSubtree.

If the value of this bit mask is M bits long and there are more than M sub-identifiers in the corresponding instance of viewSubtree, then the bit mask is extended with 1's to be the required length.

Note that when the value of this object is the zero-length string, this extension rule results in a mask of all-1's being used (i.e., no 'wild card'), and the family of view subtrees is the one view subtree uniquely identified by the corresponding instance of viewSubtree."

```
DEFVAL      { ''H }
::= { viewEntry 3 }
```

```
viewType OBJECT-TYPE
    SYNTAX      INTEGER  {
                    included(1),
                    excluded(2)
                }
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The status of a particular family of view
            subtrees within the particular SNMPv2 context's
            MIB view.  The value 'included(1)' indicates that
            the corresponding instances of viewSubtree and
            viewMask define a family of view subtrees included
            in the MIB view.  The  value 'excluded(2)'
            indicates that the corresponding instances of
            viewSubtree and viewMask define a family of view
            subtrees excluded from the MIB view."
    DEFVAL      { included }
    ::= { viewEntry 4 }

viewStorageType OBJECT-TYPE
    SYNTAX      StorageType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The storage type for this conceptual row in the
            viewTable."
    DEFVAL      { nonVolatile }
    ::= { viewEntry 5 }

viewStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The status of this conceptual row in the
            viewTable."
    ::= { viewEntry 6 }
```

```
-- conformance information

partyMIBConformance
               OBJECT IDENTIFIER ::= { partyMIB 3 }

partyMIBCompliances
               OBJECT IDENTIFIER ::= { partyMIBConformance 1 }
partyMIBGroups
               OBJECT IDENTIFIER ::= { partyMIBConformance 2 }


-- compliance statements

unSecurableCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
            "The compliance statement for SNMPv2 entities
            which implement the Party MIB, but do not support
            any authentication or privacy protocols (i.e.,
            only the noAuth and noPriv protocols are
            supported)."
    MODULE  -- this module
        MANDATORY-GROUPS { partyMIBGroup }
    ::= { partyMIBCompliances 1 }


partyNoPrivacyCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
            "The compliance statement for SNMPv2 entities
            which implement the Party MIB, and support an
            authentication protocol, but do not support any
            privacy protocols (i.e., only the noAuth,
            v2md5AuthProtocol, and noPriv protocols are
            supported)."
    MODULE  -- this module
        MANDATORY-GROUPS { partyMIBGroup }
    ::= { partyMIBCompliances 2 }
```

```
partyPrivacyCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
            "The compliance statement for SNMPv2 entities
            which implement the Party MIB, support an
            authentication protocol, and support a privacy
            protocol ONLY for the purpose of accessing
            security parameters.

            For all aclTable entries authorizing a subject
            and/or target SNMPv2 party whose privacy protocol
            is desPrivProtocol, to be used in accessing a
            SNMPv2 context, the MIB view for that SNMPv2
            context shall include only those objects
            subordinate to partyMIBObjects, or a subset
            thereof, e.g.,

                 viewSubtree = { partyMIBObjects }
                 viewMask    = ''H
                 viewType    = { included }

            Any attempt to configure an entry in the
            partyTable, the contextTable, the aclTable or the
            viewTable such that a party using the
            desPrivProtocol would be authorized for use in
            accessing objects outside of the partyMIBObjects
            subtree shall result in the appropriate error
            response (e.g., wrongValue or inconsistentValue)."
    MODULE  -- this module
        MANDATORY-GROUPS { partyMIBGroup }
    ::= { partyMIBCompliances 3 }
```

```
  fullPrivacyCompliance MODULE-COMPLIANCE
      STATUS  current
      DESCRIPTION
              "The compliance statement for SNMPv2 entities
              which implement the Party MIB, support an
              authentication protocol, and support a privacy
              protocol without restrictions on its use."
      MODULE  -- this module
          MANDATORY-GROUPS { partyMIBGroup }
      ::= { partyMIBCompliances 4 }


-- units of conformance

partyMIBGroup OBJECT-GROUP
    OBJECTS { partyIndex, partyTDomain, partyTAddress,
              partyMaxMessageSize, partyLocal,
              partyAuthProtocol, partyAuthClock,
              partyAuthPrivate, partyAuthPublic,
              partyAuthLifetime, partyPrivProtocol,
              partyPrivPrivate, partyPrivPublic,
              partyStorageType, partyStatus,
              partyCloneFrom,
              contextIndex, contextLocal,
              contextViewIndex, contextLocalEntity,
              contextLocalTime, contextStorageType,
              contextStatus, aclTarget, aclSubject,
              aclPrivileges, aclStorageType, aclStatus,
              viewMask, viewType, viewStorageType, viewStatus }
    STATUS  current
    DESCRIPTION
            "The collection of objects allowing the
            description and configuration of SNMPv2 parties.

            Note that objects which support proxy
            relationships are not included in this conformance
            group."
    ::= { partyMIBGroups 1 }

END
```

□

3. Acknowledgments

This document is based, almost entirely, on RFC 1353.

4. References

[1] Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1), International Organization for Standardization. International Standard 8824, (December, 1987).

[2] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1442, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.

[3] Galvin, J., and McCloghrie, K., "Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1445, Trusted Information Systems, Hughes LAN Systems, April 1993.

[4] Galvin, J., and McCloghrie, K., "Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1446, Trusted Information Systems, Hughes LAN Systems, April 1993.

[5] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1448, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.

[5] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Transport Mappings for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1449, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.

5. Security Considerations

Security issues are not discussed in this memo.

6. Authors' Addresses

Keith McCloghrie
Hughes LAN Systems
1225 Charleston Road
Mountain View, CA 94043
US

Phone: +1 415 966 7934
Email: kzm@hls.com

James M. Galvin
Trusted Information Systems, Inc.
3060 Washington Road, Route 97
Glenwood, MD 21738

Phone: +1 301 854-6889
EMail: galvin@tis.com

Network Working Group
Request For Comments: 1156
Obsoletes: RFC 1066

K. McCloghrie
Hughes LAN Systems
M. Rose
Performance Systems International

May 1990

Management Information Base for Network Management of TCP/IP-based internets

Table of Contents

1. Status of this Memo

This RFC is a re-release of RFC 1066, with a changed "Status of this Memo", "IAB Policy Statement", and "Introduction" sections plus a few minor typographical corrections. The technical content of the document is unchanged from RFC 1066.

This memo provides the initial version of the Management Information Base (MIB) for use with network management protocols in TCP/IP-based internets in the short-term. In particular, together with its companion memos which describe the structure of management information along with the initial network management protocol, these documents provide a simple, workable architecture and system for managing TCP/IP-based internets and in particular the Internet.

This memo specifies a Standard Protocol for the Internet community. TCP/IP implementations in the Internet which are network manageable are expected to adopt and implement this specification.

The Internet Activities Board recommends that all IP and TCP implementations be network manageable. This implies implementation of the Internet MIB (RFC-1156) and at least one of the two recommended management protocols SNMP (RFC-1157) or CMOT (RFC-1095). It should be noted that, at this time, SNMP is a full Internet standard and CMOT is a draft standard. See also the Host and Gateway Requirements RFCs for more specific information on the applicability of this standard.

Please refer to the latest edition of the "IAB Official Protocol Standards" RFC for current information on the state and status of standard Internet protocols.

Distribution of this memo is unlimited.

2. IAB Policy Statement

This MIB specification is the first edition of an evolving document defining variables needed for monitoring and control of various components of the Internet. Not all groups of defined variables are mandatory for all Internet components.

For example, the EGP group is mandatory for gateways using EGP but not for hosts which should not be running EGP. Similarly, the TCP group is mandatory for hosts running TCP but not for gateways which aren't running it. What IS mandatory, however, is that all variables of a group be supported if any element of the group is supported.

It is expected that additional MIB groups and variables will be defined over time to accommodate the monitoring and control needs of new or changing components of the Internet. The responsible working group(s) will continue to refine this specification.

3. Introduction

As reported in RFC 1052, IAB Recommendations for the Development of Internet Network Management Standards [1], the Internet Activities Board has directed the Internet Engineering Task Force (IETF) to create two new working groups in the area of network management. One group was charged with the further specification and definition of elements to be included in the Management Information Base. The other was charged with defining the modifications to the Simple Network Management Protocol (SNMP) to accommodate the short-term needs of the network vendor and operator communities. In the long-term, the use of the OSI network management framework was to be examined using the ISO CMIS/CMIP [2,3] framework as a basis. Two documents were produced to define the management information: RFC 1065, which defined the Structure of Management Information (SMI) [4], and RFC 1066, which defined the Management Information Base (MIB) [5]. Both of these documents were designed so as to be compatible with both the SNMP and the OSI network management framework.

This strategy was quite successful in the short-term: Internet-based network management technology was fielded, by both the research and commercial communities, within a few months. As a result of this, portions of the Internet community became network manageable in a timely fashion.

As reported in RFC 1109, Report of the Second Ad Hoc Network Management Review Group [6], the requirements of the SNMP and the OSI network management frameworks were more different than anticipated. As such, the requirement for compatibility between the SMI/MIB and both frameworks was suspended.

The IAB has designated the SNMP, SMI, and the initial Internet MIB to be full "Standard Protocols" with "Recommended" status. By this action, the IAB recommends that all IP and TCP implementations be network manageable and that the implementations that are network manageable are expected to adopt and implement the SMI, MIB, and SNMP.

As such, the current network management framework for TCP/IP- based internets consists of: Structure and Identification of Management Information for TCP/IP-based Internets, which describes how managed objects contained in the MIB are defined as set forth in RFC 1155 [7]; Management Information Base for Network Management of TCP/IP-based Internets, which describes the managed objects contained in the MIB as set forth in this memo; and, the Simple Network Management Protocol, which defines the protocol used to manage these objects, as set forth in RFC 1157 [8].

The IAB also urged the working groups to be "extremely sensitive to the need to keep SNMP simple," and recommends that the MIB working group take as its starting inputs the MIB definitions found in the High-Level Entity Management Systems (HEMS) RFC 1024 [9], the initial SNMP specification [10], and the CMIS/CMIP memos [11,12].

Thus, the list of managed objects defined here, has been derived by taking only those elements which are considered essential. Since such elements are essential, there is no need to allow the implementation of individual objects, to be optional. Rather, all compliant implementations will contain all applicable (see below) objects defined in this memo.

This approach of taking only the essential objects is NOT restrictive, since the SMI defined in the companion memo provides three extensibility mechanisms: one, the addition of new standard objects through the definitions of new versions of the MIB; two, the addition of widely-available but non-standard objects through the multilateral subtree; and three, the addition of private objects through the enterprises subtree. Such additional objects can not only be used for vendor-specific elements, but also for experimentation as required to further the knowledge of which other objects are essential.

The primary criterion for being considered essential was for an object to be contained in all of the above referenced MIB definitions. A few other objects have been included, but only if the MIB working group believed they are truly essential. The detailed list of criteria against which potential inclusions in this (initial) MIB were considered, was:

1) An object needed to be essential for either fault or configuration management.

2) Only weak control objects were permitted (by weak, it is meant that tampering with them can do only limited damage). This criterion reflects the fact that the current management protocols are not sufficiently secure to do more powerful control operations.

3) Evidence of current use and utility was required.

4) An attempt was made to limit the number of objects to about 100 to make it easier for vendors to fully instrument their software.

5) To avoid redundant variables, it was required that no object be included that can be derived from others in the MIB.

6) Implementation specific objects (e.g., for BSD UNIX) were excluded.

7) It was agreed to avoid heavily instrumenting critical

sections of code. The general guideline was one counter per critical section per layer.

4. Objects

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using Abstract Syntax Notation One (ASN.1) [13].

The mechanisms used for describing these objects are specified in the companion memo. In particular, each object has a name, a syntax, and an encoding. The name is an object identifier, an administratively assigned name, which specifies an object type. The object type together with an object instance serves to uniquely identify a specific instantiation of the object. For human convenience, we often use a textual string, termed the OBJECT DESCRIPTOR, to also refer to the object type.

The syntax of an object type defines the abstract data structure corresponding to that object type. The ASN.1 language is used for this purpose. However, the companion memo purposely restricts the ASN.1 constructs which may be used. These restrictions are explicitly made for simplicity.

The encoding of an object type is simply how that object type is represented using the object type's syntax. Implicitly tied to the notion of an object type's syntax and encoding is how the object type is represented when being transmitted on the network. This memo specifies the use of the basic encoding rules of ASN.1 [14].

4.1. Object Groups

Since this list of managed objects contains only the essential elements, there is no need to allow individual objects to be optional. Rather, the objects are arranged into the following groups:

- System
- Interfaces
- Address Translation
- IP
- ICMP
- TCP
- UDP
- EGP

There are two reasons for defining these groups: one, to provide a means of assigning object identifiers; two, to provide a method for implementations of managed agents to know which objects they must implement. This method is as follows: if the semantics of a group is applicable to an implementation, then it must implement all objects in that group. For example, an implementation must implement the EGP group if and only if it implements the EGP protocol.

4.2. Format of Definitions

The next section contains the specification of all object types contained in the MIB. Following the conventions of the companion memo, the object types are defined using the following fields:

OBJECT:

A textual name, termed the OBJECT DESCRIPTOR, for the object type, along with its corresponding OBJECT IDENTIFIER.

Syntax:
The abstract syntax for the object type, presented using ASN.1. This must resolve to an instance of the ASN.1 type ObjectSyntax defined in the SMI.

Definition:
A textual description of the semantics of the object type. Implementations should ensure that their interpretation of the object type fulfills this definition since this MIB is intended for use in multi-vendor environments. As such it is vital that object types have consistent meaning across all machines.

Access:
One of read-only, read-write, write-only, or not-accessible.

Status:
One of mandatory, optional, or obsolete.

5. Object Definitions

```
RFC1156-MIB

DEFINITIONS ::= BEGIN

IMPORTS
        mgmt, OBJECT-TYPE, NetworkAddress, IpAddress,
        Counter, Gauge, TimeTicks
            FROM RFC1155-SMI;

mib        OBJECT IDENTIFIER ::= { mgmt 1 }

system     OBJECT IDENTIFIER ::= { mib 1 }
interfaces OBJECT IDENTIFIER ::= { mib 2 }
at         OBJECT IDENTIFIER ::= { mib 3 }
ip         OBJECT IDENTIFIER ::= { mib 4 }
icmp       OBJECT IDENTIFIER ::= { mib 5 }
tcp        OBJECT IDENTIFIER ::= { mib 6 }
udp        OBJECT IDENTIFIER ::= { mib 7 }
egp        OBJECT IDENTIFIER ::= { mib 8 }

END
```

5.1. The System Group

Implementation of the System group is mandatory for all systems.

OBJECT:
----
sysDescr { system 1 }

Syntax:
OCTET STRING

Definition:
A textual description of the entity. This value should include the full name and version identification of the system's hardware type, software operating-system, and networking software. It is mandatory that this only contain printable ASCII characters.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
sysObjectID { system 2 }

Syntax:
OBJECT IDENTIFIER

Definition:
The vendor's authoritative identification of the network management subsystem contained in the entity. This value is allocated within the SMI enterprises subtree (1.3.6.1.4.1) and provides an easy and unambiguous means for determining "what kind of box" is being managed. For example, if vendor "Flintstones, Inc." was assigned the subtree 1.3.6.1.4.1.42, it could assign the identifier 1.3.6.1.4.1.42.1.1 to its "Fred Router".

Access:
read-only.

Status:
mandatory.

```
OBJECT:
-------
     sysUpTime { system 3 }

Syntax:
     TimeTicks

Definition:
     The time (in hundredths of a second) since the network
     management portion of the system was last re-initialized.

Access:
     read-only.

Status:
     mandatory.
```

5.2. The Interfaces Group

Implementation of the Interfaces group is mandatory for all systems.

```
OBJECT:
-------
     ifNumber { interfaces 1 }

Syntax:
     INTEGER

Definition:
     The number of network interfaces (regardless of their
     current state) on which this system can send/receive IP
     datagrams.

Access:
     read-only.

Status:
     mandatory.
```

5.2.1. The Interfaces Table

```
OBJECT:
-------
     ifTable { interfaces 2 }

Syntax:
     SEQUENCE OF IfEntry

Definition:
     A list of interface entries.  The number of entries is
     given by the value of ifNumber.

Access:
     read-write.

Status:
     mandatory.

OBJECT:
-------
     ifEntry { ifTable 1 }

Syntax:
     IfEntry ::= SEQUENCE {
```

```
            ifIndex
                INTEGER,
            ifDescr
                OCTET STRING,
            ifType
                INTEGER,
            ifMtu
                INTEGER,
            ifSpeed
                Gauge,
            ifPhysAddress
                OCTET STRING,
            ifAdminStatus
                INTEGER,
            ifOperStatus
                INTEGER,
            ifLastChange
                TimeTicks,
            ifInOctets
                Counter,
            ifInUcastPkts
                Counter,
            ifInNUcastPkts
                Counter,
            ifInDiscards
                Counter,
            ifInErrors
                Counter,
            ifInUnknownProtos
                Counter,
            ifOutOctets
                Counter,
            ifOutUcastPkts
                Counter,
            ifOutNUcastPkts
                Counter,
            ifOutDiscards
                Counter,
            ifOutErrors
                Counter,
            ifOutQLen
                Gauge
       }
```

Definition:
An interface entry containing objects at the subnetwork layer and below for a particular interface.

Access:
    read-write.

Status:
    mandatory.

We now consider the individual components of each interface entry:

OBJECT:
----
    ifIndex { ifEntry 1 }

Syntax:
    INTEGER

Definition:
    A unique value for each interface. Its value ranges between 1 and the value of ifNumber. The value for each interface must remain constant at least from one re-initialization of the entity's network management system to the next re-initialization.

Access:
    read-only.

Status:
    mandatory.

OBJECT:
----
    ifDescr { ifEntry 2 }

Syntax:
    OCTET STRING

Definition:
    A text string containing information about the interface. This string should include the name of the manufacturer, the product name and the version of the hardware interface. The string is intended for presentation to a human; it must not contain anything but printable ASCII characters.

```
Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     ifType { ifEntry 3 }

Syntax:
     INTEGER {
          other(1),           -- none of the following
          regular1822(2),
          hdh1822(3),
          ddn-x25(4),
          rfc877-x25(5),
          ethernet-csmacd(6),
          iso88023-csmacd(7),
          iso88024-tokenBus(8),
          iso88025-tokenRing(9),
          iso88026-man(10),
          starLan(11),
          proteon-10MBit(12),
          proteon-80MBit(13),
          hyperchannel(14),
          fddi(15),
          lapb(16),
          sdlc(17),
          t1-carrier(18),
          cept(19),           -- european equivalent of T-1
          basicIsdn(20),
          primaryIsdn(21),
                               -- proprietary serial
          propPointToPointSerial(22)
     }

Definition:
     The type of interface, distinguished according to the
     physical/link/network protocol(s) immediately "below" IP
     in the protocol stack.

Access:
     read-only.

Status:
     mandatory.
```

OBJECT:
----
ifMtu { ifEntry 4 }

Syntax:
INTEGER

Definition:
The size of the largest IP datagram which can be sent/received on the interface, specified in octets.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
ifSpeed { ifEntry 5 }

Syntax:
Gauge

Definition:
An estimate of the interface's current bandwidth in bits per second. For interfaces which do not vary in bandwidth or for those where no accurate estimation can be made, this object should contain the nominal bandwidth.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
ifPhysAddress { ifEntry 6 }

Syntax:
OCTET STRING

Definition:
The interface's address at the protocol layer immediately

```
          "below" IP in the protocol stack.  For interfaces which
          do not have such an address (e.g., a serial line), this
          object should contain an octet string of zero length.

     Access:
          read-only.

     Status:
          mandatory.


     OBJECT:
     -------
          ifAdminStatus { ifEntry 7 }

     Syntax:
          INTEGER {
               up(1),       -- ready to pass packets
               down(2),
               testing(3)   -- in some test mode
             }

      Definition:
          The desired state of the interface.  The testing(3) state
          indicates that no operational packets can be passed.

      Access:
          read-write.

      Status:
          mandatory.


     OBJECT:
     -------
          ifOperStatus { ifEntry 8 }

     Syntax:
          INTEGER {
               up(1),       -- ready to pass packets
               down(2),
               testing(3)   -- in some test mode
          }

     Definition:
          The current operational state of the interface.  The
          testing(3) state indicates that no operational packets
          can be passed.
```

```
Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     ifLastChange { ifEntry 9 }

Syntax:
     TimeTicks

Definition:
     The value of sysUpTime at the time the interface entered
     its current operational state.  If the current state was
     entered prior to the last re-initialization of the local
     network management subsystem, then this object contains a
     zero value.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     ifInOctets { ifEntry 10 }

Syntax:
     Counter

Definition:
     The total number of octets received on the interface,
     including framing characters.

Access:
     read-only.

Status:
     mandatory.
```

OBJECT:
----
ifInUcastPkts { ifEntry 11 }

Syntax:
Counter

Definition:
The number of (subnet) unicast packets delivered to a higher-layer protocol.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
ifInNUcastPkts { ifEntry 12 }

Syntax:
Counter

Definition:
The number of non-unicast (i.e., subnet broadcast or subnet multicast) packets delivered to a higher-layer protocol.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
ifInDiscards { ifEntry 13 }

Syntax:
Counter

Definition:
The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer

protocol. One possible reason for discarding such a packet could be to free up buffer space.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
ifInErrors { ifEntry 14 }

Syntax:
Counter

Definition:
The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
ifInUnknownProtos { ifEntry 15 }

Syntax:
Counter

Definition:
The number of packets received via the interface which were discarded because of an unknown or unsupported protocol.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
ifOutOctets { ifEntry 16 }

Syntax:
Counter

Definition:
The total number of octets transmitted out of the interface, including framing characters.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
ifOutUcastPkts { ifEntry 17 }

Syntax:
Counter

Definition:
The total number of packets that higher-level protocols requested be transmitted to a subnet-unicast address, including those that were discarded or not sent.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
ifOutNUcastPkts { ifEntry 18 }

Syntax:
Counter

Definition:
The total number of packets that higher-level protocols requested be transmitted to a non-unicast (i.e., a subnet broadcast or subnet multicast) address, including those

that were discarded or not sent.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
ifOutDiscards { ifEntry 19 }

Syntax:
Counter

Definition:
The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
ifOutErrors { ifEntry 20 }

Syntax:
Counter

Definition:
The number of outbound packets that could not be transmitted because of errors.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
ifOutQLen { ifEntry 21 }

Syntax:
Gauge

Definition:
The length of the output packet queue (in packets).

Access:
read-only.

Status:
mandatory.

5.3. The Address Translation Group

Implementation of the Address Translation group is mandatory for all systems.

The Address Translation group contains one table which is the union across all interfaces of the translation tables for converting a NetworkAddress (e.g., an IP address) into a subnetwork-specific address. For lack of a better term, this document refers to such a subnetwork-specific address as a "physical" address.

Examples of such translation tables are: for broadcast media where ARP is in use, the translation table is equivalent to the ARP cache; or, on an X.25 network where non-algorithmic translation to X.121 addresses is required, the translation table contains the NetworkAddress to X.121 address equivalences.

```
OBJECT:
-------
     atTable { at 1 }

Syntax:
     SEQUENCE OF AtEntry

Definition:
     The Address Translation tables contain the NetworkAddress
     to "physical" address equivalences.  Some interfaces do
     not use translation tables for determining address
     equivalences (e.g., DDN-X.25 has an algorithmic method);
     if all interfaces are of this type, then the Address
     Translation table is empty, i.e., has zero entries.

Access:
     read-write.

Status:
     mandatory.


OBJECT:
-------
     atEntry { atTable 1 }

Syntax:
     AtEntry ::= SEQUENCE {
          atIfIndex
```

```
                INTEGER,
            atPhysAddress
                OCTET STRING,
            atNetAddress
                NetworkAddress
       }
```

Definition:
     Each entry contains one NetworkAddress to "physical"
     address equivalence.

Access:
     read-write.

Status:
     mandatory.

We now consider the individual components of each Address
Translation table entry:

OBJECT:
----
     atIfIndex { atEntry 1 }

Syntax:
     INTEGER

Definition:
     The interface on which this entry's equivalence is
     effective. The interface identified by a particular
     value of this index is the same interface as identified
     by the same value of ifIndex.

Access:
     read-write.

Status:
     mandatory.

OBJECT:
----
     atPhysAddress { atEntry 2 }

Syntax:
     OCTET STRING

```
     Definition:
          The media-dependent "physical" address.

     Access:
          read-write.

     Status:
          mandatory.


     OBJECT:
     -------
          atNetAddress { atEntry 3 }

     Syntax:
          NetworkAddress

     Definition:
          The NetworkAddress (e.g., the IP address) corresponding to
          the media-dependent "physical" address.

     Access:
          read-write.

     Status:
          mandatory.
```

5.4. The IP Group

Implementation of the IP group is mandatory for all systems.

```
OBJECT:
-------
     ipForwarding { ip 1 }

Syntax:
     INTEGER {
          gateway(1),    -- entity forwards datagrams
          host(2)        -- entity does NOT forward datagrams
     }

Definition:
     The indication of whether this entity is acting as an IP
     gateway in respect to the forwarding of datagrams
     received by, but not addressed to, this entity.  IP
     gateways forward datagrams; Hosts do not (except those
     Source-Routed via the host).

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     ipDefaultTTL { ip 2 }

Syntax:
     INTEGER

Definition:
     The default value inserted into the Time-To-Live field of
     the IP header of datagrams originated at this entity,
     whenever a TTL value is not supplied by the transport
     layer protocol.

Access:
     read-write.

Status:
     mandatory.
```

```
OBJECT:
-------
     ipInReceives { ip 3 }

Syntax:
     Counter

Definition:
     The total number of input datagrams received from
     interfaces, including those received in error.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     ipInHdrErrors { ip 4 }

Syntax:
     Counter

Definition:
     The number of input datagrams discarded due to errors in
     their IP headers, including bad checksums, version number
     mismatch, other format errors, time-to-live exceeded,
     errors discovered in processing their IP options, etc.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     ipInAddrErrors { ip 5 }

Syntax:
     Counter

Definition:
     The number of input datagrams discarded because the IP
     address in their IP header's destination field was not a
```

valid address to be received at this entity. This count includes invalid addresses (e.g., 0.0.0.0) and addresses of unsupported Classes (e.g., Class E). For entities which are not IP Gateways and therefore do not forward datagrams, this counter includes datagrams discarded because the destination address was not a local address.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
ipForwDatagrams { ip 6 }

Syntax:
Counter

Definition:
The number of input datagrams for which this entity was not their final IP destination, as a result of which an attempt was made to find a route to forward them to that final destination. In entities which do not act as IP Gateways, this counter will include only those packets which were Source-Routed via this entity, and the Source-Route option processing was successful.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
ipInUnknownProtos { ip 7 }

Syntax:
Counter

Definition:
The number of locally-addressed datagrams received successfully but discarded because of an unknown or unsupported protocol.

```
 Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     ipInDiscards { ip 8 }

Syntax:
     Counter

Definition:
     The number of input IP datagrams for which no problems
     were encountered to prevent their continued processing,
     but which were discarded (e.g. for lack of buffer space).
     Note that this counter does not include any datagrams
     discarded while awaiting re-assembly.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     ipInDelivers { ip 9 }

Syntax:
     Counter

Definition:
     The total number of input datagrams successfully
     delivered to IP user-protocols (including ICMP).

Access:
     read-only.

Status:
     mandatory.

OBJECT:
-------
     ipOutRequests { ip 10 }
```

Syntax:
Counter

Definition:
The total number of IP datagrams which local IP user-protocols (including ICMP) supplied to IP in requests for transmission. Note that this counter does not include any datagrams counted in ipForwDatagrams.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
ipOutDiscards { ip 11 }

Syntax:
Counter

Definition:
The number of output IP datagrams for which no problem was encountered to prevent their transmission to their destination, but which were discarded (e.g., for lack of buffer space). Note that this counter would include datagrams counted in ipForwDatagrams if any such packets met this (discretionary) discard criterion.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
ipOutNoRoutes { ip 12 }

Syntax:
Counter

Definition:
The number of IP datagrams discarded because no route could be found to transmit them to their destination. Note that this counter includes any packets counted in ipForwDatagrams which meet this "no-route" criterion.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
ipReasmTimeout { ip 13 }

Syntax:
INTEGER

Definition:
The maximum number of seconds which received fragments are held while they are awaiting reassembly at this entity.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
ipReasmReqds { ip 14 }

Syntax:
Counter

Definition:
The number of IP fragments received which needed to be reassembled at this entity.

Access:
read-only.

Status:
mandatory.

```
OBJECT:
-------
     ipReasmOKs { ip 15 }

Syntax:
     Counter

Definition:
     The number of IP datagrams successfully re-assembled.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     ipReasmFails { ip 16 }

Syntax:
     Counter

Definition:
     The number of failures detected by the IP re-assembly
     algorithm (for whatever reason:  timed out, errors, etc).

     Note that this is not necessarily a count of discarded IP
     fragments since some algorithms (notably RFC 815's) can
     lose track of the number of fragments by combining them
     as they are received.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     ipFragOKs { ip 17 }

Syntax:
    Counter
```

```
Definition:
     The number of IP datagrams that have been successfully
     fragmented at this entity.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     ipFragFails { ip 18 }

Syntax:
     Counter

Definition:
     The number of IP datagrams that have been discarded
     because they needed to be fragmented at this entity but
     could not be, e.g., because their "Don't Fragment" flag
     was set.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     ipFragCreates { ip 19 }

Syntax:
     Counter

Definition:
     The number of IP datagram fragments that have been
     generated as a result of fragmentation at this entity.

Access:
     read-only.

Status:
     mandatory.
```

5.4.1. The IP Address Table

The Ip Address table contains this entity's IP addressing information.

```
OBJECT:
-------
     ipAddrTable { ip 20 }

Syntax:
     SEQUENCE OF IpAddrEntry

Definition:
     The table of addressing information relevant to this
     entity's IP addresses.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     ipAddrEntry { ipAddrTable 1 }

Syntax:
     IpAddrEntry ::= SEQUENCE {
          ipAdEntAddr
              IpAddress,
          ipAdEntIfIndex
              INTEGER,
          ipAdEntNetMask
              IpAddress,
          ipAdEntBcastAddr
              INTEGER
     }

Definition:
     The addressing information for one of this entity's IP
     addresses.

Access:
     read-only.
```

```
Status:
     mandatory.


OBJECT:
-------
     ipAdEntAddr  { ipAddrEntry 1 }

Syntax:
     IpAddress

Definition:
     The IP address to which this entry's addressing
     information pertains.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     ipAdEntIfIndex  { ipAddrEntry 2 }

Syntax:
     INTEGER

Definition:
     The index value which uniquely identifies the interface
     to which this entry is applicable.  The interface
     identified by a particular value of this index is the
      same interface as identified by the same value of
      ifIndex.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     ipAdEntNetMask  { ipAddrEntry 3 }
```

```
     Syntax:
          IpAddress

     Definition:
          The subnet mask associated with the IP address of this
          entry.  The value of the mask is an IP address with all
          the network bits set to 1 and all the hosts bits set to
          0.

     Access:
          read-only.

     Status:
          mandatory.


     OBJECT:
     -------
          ipAdEntBcastAddr { ipAddrEntry 4 }

     Syntax:
          INTEGER

     Definition:
          The value of the least-significant bit in the IP
          broadcast address used for sending datagrams on the
          (logical) interface associated with the IP address of
          this entry.  For example, when the Internet standard
          all-ones broadcast address is used, the value will be 1.

     Access:
          read-only.

     Status:
          mandatory.
```

5.4.2. The IP Routing Table

The IP Routing Table contains an entry for each route presently known to this entity. Note that the action to be taken in response to a request to read a non-existent entry, is specific to the network management protocol being used.

```
     OBJECT:
     -------
          ipRoutingTable { ip 21 }
```

```
Syntax:
     SEQUENCE OF IpRouteEntry

Definition:
     This entity's IP Routing table.

Access:
     read-write.

Status:
     mandatory.


OBJECT:
-------
     ipRouteEntry { ipRoutingTable 1 }

Syntax:
     IpRouteEntry ::= SEQUENCE {
          ipRouteDest
              IpAddress,
          ipRouteIfIndex
              INTEGER,
          ipRouteMetric1
              INTEGER,
          ipRouteMetric2
              INTEGER,
          ipRouteMetric3
              INTEGER,
          ipRouteMetric4
              INTEGER,
          ipRouteNextHop
              IpAddress,
          ipRouteType
              INTEGER,
          ipRouteProto
              INTEGER,
          ipRouteAge
              INTEGER
     }

Definition:
     A route to a particular destination.

Access:
     read-write.
```

Status:
mandatory.

We now consider the individual components of each route in the IP Routing Table:

OBJECT:
----
ipRouteDest { ipRouteEntry 1 }

Syntax:
IpAddress

Definition:
The destination IP address of this route. An entry with a value of 0.0.0.0 is considered a default route. Multiple such default routes can appear in the table, but access to such multiple entries is dependent on the table-access mechanisms defined by the network management protocol in use.

Access:
read-write.

Status:
mandatory.

OBJECT:
----
ipRouteIfIndex { ipRouteEntry 2 }

Syntax:
INTEGER

Definition:
The index value which uniquely identifies the local interface through which the next hop of this route should be reached. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.

Access:
read-write.

Status:
mandatory.

```
OBJECT:
-------
     ipRouteMetric1 { ipRouteEntry 3 }

Syntax:
     INTEGER

Definition:
     The primary routing metric for this route.  The semantics
     of this metric are determined by the routing-protocol
     specified in the route's ipRouteProto value.  If this
     metric is not used, its value should be set to -1.

Access:
     read-write.

Status:
     mandatory.


OBJECT:
-------
     ipRouteMetric2 { ipRouteEntry 4 }

Syntax:
     INTEGER

Definition:
     An alternate routing metric for this route.  The
     semantics of this metric are determined by the routing-
     protocol specified in the route's ipRouteProto value.  If
     this metric is not used, its value should be set to -1.

Access:
     read-write.

Status:
     mandatory.


OBJECT:
-------
     ipRouteMetric3 { ipRouteEntry 5 }

Syntax:
     INTEGER
```

Definition:
An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

Access:
read-write.

Status:
mandatory.

OBJECT:
----
ipRouteMetric4 { ipRouteEntry 6 }

Syntax:
INTEGER

Definition:
An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

Access:
read-write.

Status:
mandatory.

OBJECT:
----
ipRouteNextHop { ipRouteEntry 7 }

Syntax:
IpAddress

Definition:
The IP address of the next hop of this route.

Access:
read-write.

Status:
mandatory.

```
OBJECT:
-------
     ipRouteType { ipRouteEntry 8 }

Syntax:
     INTEGER {
          other(1),         -- none of the following

          invalid(2),       -- an invalidated route

                            -- route to directly
          direct(3),        -- connected (sub-)network

                            -- route to a non-local
          remote(4),        -- host/network/sub-network
     }

Definition:
     The type of route.

Access:
     read-write.

Status:
     mandatory.


OBJECT:
-------
     ipRouteProto { ipRouteEntry 9 }

Syntax:
     INTEGER {
          other(1),        -- none of the following

                           -- non-protocol information,
                           -- e.g., manually configured
          local(2),        -- entries

                           -- set via a network management
          netmgmt(3),      -- protocol

                           -- obtained via ICMP,
          icmp(4),         -- e.g., Redirect

                           -- the remaining values are
                           -- all gateway routing protocols
          egp(5),
```

```
                  ggp(6),
                  hello(7),
                  rip(8),
                  is-is(9),
                  es-is(10),
                  ciscoIgrp(11),
                  bbnSpfIgp(12),
                  oigp(13)
             }
```

Definition:
     The routing mechanism via which this route was learned. Inclusion of values for gateway routing protocols is not intended to imply that hosts should support those protocols.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     ipRouteAge { ipRouteEntry 10 }

Syntax:
     INTEGER

Definition:
     The number of seconds since this route was last updated or otherwise determined to be correct. Note that no semantics of "too old" can be implied except through knowledge of the routing protocol by which the route was learned.

Access:
     read-write.

Status:
     mandatory.

5.5. The ICMP Group

Implementation of the ICMP group is mandatory for all systems.

The ICMP group contains the ICMP input and output statistics.

Note that individual counters for ICMP message (sub-)codes have been omitted from this (version of the) MIB for simplicity.

```
OBJECT:
-------
     icmpInMsgs { icmp 1 }

Syntax:
     Counter

Definition:
     The total number of ICMP messages which the entity
     received.  Note that this counter includes all those
     counted by icmpInErrors.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     icmpInErrors { icmp 2 }

Syntax:
     Counter

Definition:
     The number of ICMP messages which the entity received but
     determined as having errors (bad ICMP checksums, bad
     length, etc.).

Access:
     read-only.

Status:
     mandatory.
```

```
OBJECT:
-------
    icmpInDestUnreachs { icmp 3 }

Syntax:
    Counter

Definition:
     The number of ICMP Destination Unreachable messages
     received.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     icmpInTimeExcds { icmp 4 }

Syntax:
     Counter

Definition:
     The number of ICMP Time Exceeded messages received.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     icmpInParmProbs { icmp 5 }

Syntax:
     Counter

Definition:
     The number of ICMP Parameter Problem messages received.

Access:
     read-only.
```

Status:
mandatory.

OBJECT:
----
icmpInSrcQuenchs { icmp 6 }

Syntax:
Counter

Definition:
The number of ICMP Source Quench messages received.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
icmpInRedirects { icmp 7 }

Syntax:
Counter

Definition:
The number of ICMP Redirect messages received.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
icmpInEchos { icmp 8 }

Syntax:
Counter

Definition:
The number of ICMP Echo (request) messages received.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
icmpInEchoReps { icmp 9 }

Syntax:
Counter

Definition:
The number of ICMP Echo Reply messages received.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
icmpInTimestamps { icmp 10 }

Syntax:
Counter

Definition:
The number of ICMP Timestamp (request) messages received.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
icmpInTimestampReps { icmp 11 }

Syntax:
Counter

Definition:
The number of ICMP Timestamp Reply messages received.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
icmpInAddrMasks { icmp 12 }

Syntax:
Counter

Definition:
The number of ICMP Address Mask Request messages received.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
icmpInAddrMaskReps { icmp 13 }

Syntax:
Counter

Definition:
The number of ICMP Address Mask Reply messages received.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
icmpOutMsgs { icmp 14 }

Syntax:
    Counter

Definition:
    The total number of ICMP messages which this entity attempted to send. Note that this counter includes all those counted by icmpOutErrors.

Access:
    read-only.

Status:
    mandatory.

OBJECT:
----
    icmpOutErrors { icmp 15 }

Syntax:
    Counter

Definition:
    The number of ICMP messages which this entity did not send due to problems discovered within ICMP such as a lack of buffers. This value should not include errors discovered outside the ICMP layer such as the inability of IP to route the resultant datagram. In some implementations there may be no types of error which contribute to this counter's value.

Access:
    read-only.

Status:
    mandatory.

OBJECT:
----
    icmpOutDestUnreachs { icmp 16 }

Syntax:
    Counter

Definition:
    The number of ICMP Destination Unreachable messages sent.

```
          Access:
               read-only.

          Status:
               mandatory.


          OBJECT:
          -------
               icmpOutTimeExcds { icmp 17 }

          Syntax:
               Counter

          Definition:
               The number of ICMP Time Exceeded messages sent.

          Access:
               read-only.

          Status:
               mandatory.


          OBJECT:
          -------
               icmpOutParmProbs { icmp 18 }

          Syntax:
               Counter

          Definition:
               The number of ICMP Parameter Problem messages sent.

          Access:
               read-only.

          Status:
               mandatory.


          OBJECT:
          -------
               icmpOutSrcQuenchs { icmp 19 }

          Syntax:
               Counter
```

```
Definition:
     The number of ICMP Source Quench messages sent.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     icmpOutRedirects { icmp 20 }

Syntax:
     Counter

Definition:
     The number of ICMP Redirect messages sent.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     icmpOutEchos { icmp 21 }

Syntax:
     Counter

Definition:
     The number of ICMP Echo (request) messages sent.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     icmpOutEchoReps { icmp 22 }
```

Syntax:
    Counter

Definition:
    The number of ICMP Echo Reply messages sent.

Access:
    read-only.

Status:
    mandatory.

OBJECT:
----
    icmpOutTimestamps { icmp 23 }

Syntax:
    Counter

Definition:
    The number of ICMP Timestamp (request) messages sent.

Access:
    read-only.

Status:
    mandatory.

OBJECT:
----
    icmpOutTimestampReps { icmp 24 }

Syntax:
    Counter

Definition:
    The number of ICMP Timestamp Reply messages sent.

Access:
    read-only.

Status:
    mandatory.

OBJECT:
----
icmpOutAddrMasks { icmp 25 }

Syntax:
Counter

Definition:
The number of ICMP Address Mask Request messages sent.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
icmpOutAddrMaskReps { icmp 26 }

Syntax:
Counter

Definition:
The number of ICMP Address Mask Reply messages sent.

Access:
read-only.

Status:
mandatory.

5.6. The TCP Group

Implementation of the TCP group is mandatory for all systems that implement the TCP protocol.

Note that instances of object types that represent information about a particular TCP connection are transient; they persist only as long as the connection in question.

```
OBJECT:
-------
     tcpRtoAlgorithm { tcp 1 }

Syntax:
     INTEGER {
          other(1),     -- none of the following
          constant(2),  -- a constant rto
          rsre(3),      -- MIL-STD-1778, Appendix B
          vanj(4)       -- Van Jacobson's algorithm [15]
     }

Definition:
     The algorithm used to determine the timeout value used
     for retransmitting unacknowledged octets.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     tcpRtoMin { tcp 2 }

Syntax:
     INTEGER

Definition:
     The minimum value permitted by a TCP implementation
     for the retransmission timeout, measured in
     milliseconds.  More refined semantics for objects
     of this type depend upon the algorithm used to
     determine the retransmission timeout.  In particular,
     when the timeout algorithm is rsre(3), an object
     of this type has the semantics of the LBOUND
     quantity described in RFC 793.
```

```
Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     tcpRtoMax { tcp 3 }

Syntax:
     INTEGER

Definition:
     The maximum value permitted by a TCP implementation
     for the retransmission timeout, measured
     in milliseconds.  More refined semantics for objects
     of this type depend upon the algorithm used to
     determine the retransmission timeout.  In particular,
     when the timeout algorithm is rsre(3), an object of
     this type has the semantics of the UBOUND quantity
     described in RFC 793.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     tcpMaxConn { tcp 4 }

Syntax:
     INTEGER

Definition:
     The limit on the total number of TCP connections the
     entity can support.  In entities where the maximum
     number of connections is dynamic, this object should
     contain the value "-1".

Access:
     read-only.
```

```
Status:
     mandatory.


OBJECT:
-------
     tcpActiveOpens { tcp 5 }

Syntax:
     Counter

Definition:
     The number of times TCP connections have made a direct
     transition to the SYN-SENT state from the CLOSED
     state.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     tcpPassiveOpens { tcp 6 }

Syntax:
     Counter

Definition:
     The number of times TCP connections have made a direct
     transition to the SYN-RCVD state from the LISTEN
     state.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     tcpAttemptFails { tcp 7 }

Syntax:
     Counter
```

```
Definition:
     The number of times TCP connections have made a direct
     transition to the CLOSED state from either the
     SYN-SENT state or the SYN-RCVD state, plus the number
     of times TCP connections have made a direct transition
     to the LISTEN state from the SYN-RCVD state.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     tcpEstabResets { tcp 8 }

Syntax:
     Counter

Definition:
     The number of times TCP connections have made a direct
     transition to the CLOSED state from either the
     ESTABLISHED state or the CLOSE-WAIT state.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     tcpCurrEstab { tcp 9 }

Syntax:
     Gauge

Definition:
     The number of TCP connections for which the current
     state is either ESTABLISHED or CLOSE-WAIT.

Access:
     read-only.
```

Status:
mandatory.

OBJECT:
----
tcpInSegs { tcp 10 }

Syntax:
Counter

Definition:
The total number of segments received, including those received in error. This count includes segments received on currently established connections.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
tcpOutSegs { tcp 11 }

Syntax:
Counter

Definition:
The total number of segments sent, including those on current connections but excluding those containing only retransmitted octets.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
tcpRetransSegs { tcp 12 }

Syntax:
Counter

```
Definition:
     The total number of segments retransmitted - that is,
     the number of TCP segments transmitted containing one
     or more previously transmitted octets.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     tcpConnTable { tcp 13 }

Syntax:
     SEQUENCE OF TcpConnEntry

Definition:
     A table containing TCP connection-specific
     information.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     tcpConnEntry { tcpConnTable 1 }

Syntax:
     TcpConnEntry ::= SEQUENCE {
          tcpConnState
              INTEGER,
          tcpConnLocalAddress
              IpAddress,
          tcpConnLocalPort
              INTEGER (0..65535),
          tcpConnRemAddress
              IpAddress,
          tcpConnRemPort
              INTEGER (0..65535)
     }
```

```
Definition:
     Information about a particular current TCP connection.
     An object of this type is transient, in that it ceases
     to exist when (or soon after) the connection makes the
     transition to the CLOSED state.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     tcpConnState { tcpConnEntry 1 }

Syntax:
     INTEGER {
          closed(1),
          listen(2),
          synSent(3),
          synReceived(4),
          established(5),
          finWait1(6),
          finWait2(7),
          closeWait(8),
          lastAck(9),
          closing(10),
          timeWait(11)
     }

Definition:
     The state of this TCP connection.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     tcpConnLocalAddress { tcpConnEntry 2 }

Syntax:
     IpAddress
```

```
Definition:
     The local IP address for this TCP connection.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     tcpConnLocalPort { tcpConnEntry 3 }

Syntax:
     INTEGER (0..65535)

Definition:
     The local port number for this TCP connection.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     tcpConnRemAddress { tcpConnEntry 4 }

Syntax:
     IpAddress

Definition:
     The remote IP address for this TCP connection.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     tcpConnRemPort { tcpConnEntry 5 }
```

Syntax:
INTEGER (0..65535)

Definition:
The remote port number for this TCP connection.

Access:
read-only.

Status:
mandatory.

5.7. The UDP Group

Implementation of the UDP group is mandatory for all systems which implement the UDP protocol.

```
OBJECT:
-------
     udpInDatagrams { udp 1 }

Syntax:
     Counter

Definition:
     The total number of UDP datagrams delivered to UDP
     users.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     udpNoPorts { udp 2 }

Syntax:
     Counter

Definition:
     The total number of received UDP datagrams for which
     there was no application at the destination port.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     udpInErrors { udp 3 }

Syntax:
     Counter
```

Definition:
The number of received UDP datagrams that could not be delivered for reasons other than the lack of an application at the destination port.

Access:
read-only.

Status:
mandatory.

OBJECT:
----
udpOutDatagrams { udp 4 }

Syntax:
Counter

Definition:
The total number of UDP datagrams sent from this entity.

Access:
read-only.

Status:
mandatory.

5.8. The EGP Group

Implementation of the EGP group is mandatory for all systems which implement the EGP protocol.

```
OBJECT:
-------
     egpInMsgs { egp 1 }

Syntax:
     Counter

Definition:
     The number of EGP messages received without error.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     egpInErrors { egp 2 }

Syntax:
     Counter

Definition:
     The number of EGP messages received that proved to be
     in error.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     egpOutMsgs { egp 3 }

Syntax:
     Counter
```

```
Definition:
     The total number of locally generated EGP messages.

Access:
     read-only.

Status:
     mandatory.


OBJECT:
-------
     egpOutErrors { egp 4 }

Syntax:
     Counter

Definition:
     The number of locally generated EGP messages not sent
     due to resource limitations within an EGP entity.

Access:
     read-only.

Status:
     mandatory.
```

5.8.1. The EGP Neighbor Table

The Egp Neighbor table contains information about this entity's EGP neighbors.

```
OBJECT:
-------
     egpNeighTable { egp 5 }

Syntax:
     SEQUENCE OF EgpNeighEntry

Definition:
     The EGP neighbor table.

Access:
     read-only.

Status:
     mandatory.
```

```
OBJECT:
-------
     egpNeighEntry { egpNeighTable 1 }

Syntax:
     EgpNeighEntry ::= SEQUENCE {
          egpNeighState
              INTEGER,
          egpNeighAddr
              IpAddress
     }

Definition:
     Information about this entity's relationship with a
     particular EGP neighbor.

Access:
     read-only.

Status:
     mandatory.


We now consider the individual components of each EGP
neighbor entry:


OBJECT:
-------
     egpNeighState { egpNeighEntry 1 }

Syntax:
     INTEGER {
          idle(1),
          acquisition(2),
          down(3),
          up(4),
          cease(5)
     }

Definition:
     The EGP state of the local system with respect to this
     entry's EGP neighbor.  Each EGP state is represented
     by a value that is one greater than the numerical
     value associated with said state in RFC 904.

Access:
     read-only.
```

Status:
    mandatory.

OBJECT:
----
    egpNeighAddr { egpNeighEntry 2 }

Syntax:
    IpAddress

Definition:
    The IP address of this entry's EGP neighbor.

Access:
    read-only.

Status:
    mandatory.

6. Definitions

```
RFC1156-MIB

DEFINITIONS ::= BEGIN

IMPORTS
        mgmt, OBJECT-TYPE, NetworkAddress, IpAddress,
        Counter, Gauge, TimeTicks
            FROM RFC1155-SMI;

  mib        OBJECT IDENTIFIER ::= { mgmt 1 }

  system     OBJECT IDENTIFIER ::= { mib 1 }
  interfaces OBJECT IDENTIFIER ::= { mib 2 }
  at         OBJECT IDENTIFIER ::= { mib 3 }
  ip         OBJECT IDENTIFIER ::= { mib 4 }
  icmp       OBJECT IDENTIFIER ::= { mib 5 }
  tcp        OBJECT IDENTIFIER ::= { mib 6 }
  udp        OBJECT IDENTIFIER ::= { mib 7 }
  egp        OBJECT IDENTIFIER ::= { mib 8 }

  -- object types

  -- the System group

  sysDescr OBJECT-TYPE
          SYNTAX  OCTET STRING
          ACCESS  read-only
          STATUS  mandatory
          ::= { system 1 }

  sysObjectID OBJECT-TYPE
          SYNTAX  OBJECT IDENTIFIER
          ACCESS  read-only
          STATUS  mandatory
          ::= { system 2 }

  sysUpTime OBJECT-TYPE
          SYNTAX  TimeTicks
          ACCESS  read-only
          STATUS  mandatory
          ::= { system 3 }

  -- the Interfaces group

  ifNumber OBJECT-TYPE
          SYNTAX  INTEGER
```

```
        ACCESS  read-only
        STATUS  mandatory
        ::= { interfaces 1 }

-- the Interfaces table

ifTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF IfEntry
        ACCESS  read-write
        STATUS  mandatory
        ::= { interfaces 2 }

ifEntry OBJECT-TYPE
        SYNTAX  IfEntry
        ACCESS  read-write
        STATUS  mandatory
        ::= { ifTable 1 }

IfEntry ::= SEQUENCE {
    ifIndex
        INTEGER,
    ifDescr
        OCTET STRING,
    ifType
        INTEGER,
    ifMtu
        INTEGER,
    ifSpeed
        Gauge,
    ifPhysAddress
        OCTET STRING,
    ifAdminStatus
        INTEGER,
    ifOperStatus
        INTEGER,
    ifLastChange
        TimeTicks,
    ifInOctets
        Counter,
    ifInUcastPkts
        Counter,
    ifInNUcastPkts
        Counter,
    ifInDiscards
        Counter,
    ifInErrors
        Counter,
    ifInUnknownProtos
```

```
            Counter,
        ifOutOctets
            Counter,
        ifOutUcastPkts
            Counter,
        ifOutNUcastPkts
            Counter,
        ifOutDiscards
            Counter,
        ifOutErrors
            Counter,
        ifOutQLen
            Gauge
    }

    ifIndex OBJECT-TYPE
            SYNTAX  INTEGER
            ACCESS  read-only
            STATUS  mandatory
            ::= { ifEntry 1 }

    ifDescr OBJECT-TYPE
            SYNTAX  OCTET STRING
            ACCESS  read-only
            STATUS  mandatory
            ::= { ifEntry 2 }

    ifType OBJECT-TYPE
            SYNTAX  INTEGER {
                    other(1),      -- none of the following
                    regular1822(2),
                    hdh1822(3),
                    ddn-x25(4),
                    rfc877-x25(5),
                    ethernet-csmacd(6),
                    iso88023-csmacd(7),
                    iso88024-tokenBus(8),
                    iso88025-tokenRing(9),
                    iso88026-man(10),
                    starLan(11),
                    proteon-10MBit(12),
                    proteon-80MBit(13),
                    hyperchannel(14),
                    fddi(15),
                    lapb(16),
                    sdlc(17),
                    t1-carrier(18),
                    cept(19),
```

```
                    basicIsdn(20),
                    primaryIsdn(21),
                                     -- proprietary serial
                    propPointToPointSerial(22)
                }
            ACCESS  read-only
            STATUS  mandatory
            ::= { ifEntry 3 }

    ifMtu OBJECT-TYPE
            SYNTAX  INTEGER
            ACCESS  read-only
            STATUS  mandatory
            ::= { ifEntry 4 }

    ifSpeed OBJECT-TYPE
            SYNTAX  Gauge
            ACCESS  read-only
            STATUS  mandatory
            ::= { ifEntry 5 }

    ifPhysAddress OBJECT-TYPE
            SYNTAX  OCTET STRING
            ACCESS  read-only
            STATUS  mandatory
            ::= { ifEntry 6 }

    ifAdminStatus OBJECT-TYPE
            SYNTAX  INTEGER {
                    up(1),          -- ready to pass packets
                    down(2),
                    testing(3)      -- in some test mode
                    }
            ACCESS  read-write
            STATUS  mandatory
            ::= { ifEntry 7 }

    ifOperStatus OBJECT-TYPE
            SYNTAX  INTEGER {
                    up(1),          -- ready to pass packets
                    down(2),
                    testing(3)      -- in some test mode
                    }
            ACCESS  read-only
            STATUS  mandatory
            ::= { ifEntry 8 }

    ifLastChange OBJECT-TYPE
```

```
                SYNTAX  TimeTicks
                ACCESS  read-only
                STATUS  mandatory
                ::= { ifEntry 9 }

        ifInOctets OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { ifEntry 10 }

        ifInUcastPkts OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::=  { ifEntry 11 }

        ifInNUcastPkts OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { ifEntry 12 }

        ifInDiscards OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { ifEntry 13 }

        ifInErrors OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { ifEntry 14 }

        ifInUnknownProtos OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { ifEntry 15 }

        ifOutOctets OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { ifEntry 16 }

        ifOutUcastPkts OBJECT-TYPE
```

```
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { ifEntry 17 }

        ifOutNUcastPkts OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { ifEntry 18 }

        ifOutDiscards OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { ifEntry 19 }

        ifOutErrors OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { ifEntry 20 }

        ifOutQLen OBJECT-TYPE
                SYNTAX  Gauge
                ACCESS  read-only
                STATUS  mandatory
                ::= { ifEntry 21 }

        -- the Address Translation group

        atTable OBJECT-TYPE
                SYNTAX  SEQUENCE OF AtEntry
                ACCESS  read-write
                STATUS  mandatory
                ::= { at 1 }

        atEntry OBJECT-TYPE
                SYNTAX  AtEntry
                ACCESS  read-write
                STATUS  mandatory
                ::= { atTable 1 }

        AtEntry ::= SEQUENCE {
            atIfIndex
                INTEGER,
            atPhysAddress
                OCTET STRING,
```

```
        atNetAddress
            NetworkAddress
    }

atIfIndex OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-write
        STATUS  mandatory
        ::= { atEntry 1 }

atPhysAddress OBJECT-TYPE
        SYNTAX  OCTET STRING
        ACCESS  read-write
        STATUS  mandatory
        ::= { atEntry 2 }

atNetAddress OBJECT-TYPE
        SYNTAX  NetworkAddress
        ACCESS  read-write
        STATUS  mandatory
        ::= { atEntry 3 }

-- the IP group

ipForwarding OBJECT-TYPE
        SYNTAX  INTEGER {
      gateway(1), -- entity forwards datagrams
      host(2)     -- entity does NOT forward datagrams
                }
        ACCESS  read-only
        STATUS  mandatory
        ::= { ip 1 }

ipDefaultTTL OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-write
        STATUS  mandatory
        ::= { ip 2 }

ipInReceives OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { ip 3 }

ipInHdrErrors OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
```

```
        STATUS  mandatory
        ::= { ip 4 }

ipInAddrErrors OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { ip 5 }

ipForwDatagrams OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { ip 6 }

ipInUnknownProtos OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { ip 7 }

ipInDiscards OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { ip 8 }

ipInDelivers OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { ip 9 }

ipOutRequests OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { ip 10 }

ipOutDiscards OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { ip 11 }

ipOutNoRoutes OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
```

```
        STATUS  mandatory
        ::= { ip 12 }

ipReasmTimeout OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        ::= { ip 13 }

ipReasmReqds OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { ip 14 }

ipReasmOKs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { ip 15 }

ipReasmFails OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { ip 16 }

ipFragOKs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { ip 17 }

ipFragFails OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { ip 18 }

ipFragCreates OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { ip 19 }

-- the IP Interface table

ipAddrTable OBJECT-TYPE
```

```
        SYNTAX  SEQUENCE OF IpAddrEntry
        ACCESS  read-only
        STATUS  mandatory
        ::= { ip 20 }

ipAddrEntry OBJECT-TYPE
        SYNTAX  IpAddrEntry
        ACCESS  read-only
        STATUS  mandatory
        ::= { ipAddrTable 1 }

IpAddrEntry ::= SEQUENCE {
    ipAdEntAddr
        IpAddress,
    ipAdEntIfIndex
        INTEGER,
    ipAdEntNetMask
        IpAddress,
    ipAdEntBcastAddr
        INTEGER
}

ipAdEntAddr OBJECT-TYPE
        SYNTAX  IpAddress
        ACCESS  read-only
        STATUS  mandatory
        ::=  { ipAddrEntry 1 }

ipAdEntIfIndex OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        ::=  { ipAddrEntry 2 }

ipAdEntNetMask OBJECT-TYPE
        SYNTAX  IpAddress
        ACCESS  read-only
        STATUS  mandatory
        ::=  { ipAddrEntry 3 }

ipAdEntBcastAddr OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        ::= { ipAddrEntry 4 }
```

```
-- the IP Routing table

ipRoutingTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF IpRouteEntry
        ACCESS  read-write
        STATUS  mandatory
        ::= { ip 21 }

ipRouteEntry OBJECT-TYPE
        SYNTAX  IpRouteEntry
        ACCESS  read-write
        STATUS  mandatory
        ::= { ipRoutingTable 1 }

IpRouteEntry ::= SEQUENCE {
    ipRouteDest
        IpAddress,
    ipRouteIfIndex
        INTEGER,
    ipRouteMetric1
        INTEGER,
    ipRouteMetric2
        INTEGER,
    ipRouteMetric3
        INTEGER,
    ipRouteMetric4
        INTEGER,
    ipRouteNextHop
        IpAddress,
    ipRouteType
        INTEGER,
    ipRouteProto
        INTEGER,
    ipRouteAge
        INTEGER
}

ipRouteDest OBJECT-TYPE
        SYNTAX  IpAddress
        ACCESS  read-write
        STATUS  mandatory
        ::= { ipRouteEntry 1 }

ipRouteIfIndex  OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-write
        STATUS  mandatory
        ::= { ipRouteEntry 2 }
```

```
ipRouteMetric1 OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-write
        STATUS  mandatory
        ::= { ipRouteEntry 3 }

ipRouteMetric2 OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-write
        STATUS  mandatory
        ::= { ipRouteEntry 4 }

ipRouteMetric3 OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-write
        STATUS  mandatory
        ::= { ipRouteEntry 5 }

ipRouteMetric4 OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-write
        STATUS  mandatory
        ::= { ipRouteEntry 6 }

ipRouteNextHop OBJECT-TYPE
        SYNTAX  IpAddress
        ACCESS  read-write
        STATUS  mandatory
        ::= { ipRouteEntry 7 }

ipRouteType OBJECT-TYPE
        SYNTAX  INTEGER {
          other(1),       -- none of the following

          invalid(2),     -- an invalidated route

                          -- route to directly
          direct(3),      -- connected (sub-)network

                          -- route to a non-local
          remote(4),      -- host/network/sub-network
            }
        ACCESS  read-write
        STATUS  mandatory
        ::= { ipRouteEntry 8 }

ipRouteProto OBJECT-TYPE
        SYNTAX  INTEGER {
```

```
                  other(1),       -- none of the following

                                  -- non-protocol information
                                  --   e.g., manually
                  local(2),       --   configured entries

                                  -- set via a network
                  netmgmt(3),     --   management protocol

                                  -- obtained via ICMP,
                  icmp(4),        --   e.g., Redirect

                                  -- the following are
                                  -- gateway routing protocols
                  egp(5),
                  ggp(6),
                  hello(7),
                  rip(8),
                  is-is(9),
                  es-is(10),
                  ciscoIgrp(11),
                  bbnSpfIgp(12),
                  oigp(13)
                    }
                ACCESS  read-only
                STATUS  mandatory
                ::= { ipRouteEntry 9 }

        ipRouteAge OBJECT-TYPE
                SYNTAX  INTEGER
                ACCESS  read-write
                STATUS  mandatory
                ::= { ipRouteEntry 10 }

        -- the ICMP group

        icmpInMsgs OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { icmp 1 }

        icmpInErrors OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { icmp 2 }
```

```
icmpInDestUnreachs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 3 }

icmpInTimeExcds OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 4 }

icmpInParmProbs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 5 }

icmpInSrcQuenchs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 6 }

icmpInRedirects OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 7 }

icmpInEchos OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 8 }

icmpInEchoReps OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 9 }

icmpInTimestamps OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 10 }
```

```
icmpInTimestampReps OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 11 }

icmpInAddrMasks OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 12 }

icmpInAddrMaskReps OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 13 }

icmpOutMsgs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 14 }

icmpOutErrors OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 15 }

icmpOutDestUnreachs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 16 }

icmpOutTimeExcds OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 17 }

icmpOutParmProbs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 18 }
```

```
icmpOutSrcQuenchs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 19 }

icmpOutRedirects OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 20 }

icmpOutEchos OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 21 }

icmpOutEchoReps OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 22 }

icmpOutTimestamps OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 23 }

icmpOutTimestampReps OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 24 }

icmpOutAddrMasks OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 25 }

icmpOutAddrMaskReps OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { icmp 26 }
```

```
-- the TCP group

tcpRtoAlgorithm OBJECT-TYPE
        SYNTAX  INTEGER {
        other(1),      -- none of the following
        constant(2),  -- a constant rto
        rsre(3),       -- MIL-STD-1778, Appendix B
        vanj(4)        -- Van Jacobson's algorithm [15]
                }
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcp 1 }

tcpRtoMin OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcp 2 }

tcpRtoMax OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcp 3 }

tcpMaxConn OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcp 4 }

tcpActiveOpens OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcp 5 }

tcpPassiveOpens OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcp 6 }

tcpAttemptFails OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcp 7 }
```

```
tcpEstabResets OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcp 8 }

tcpCurrEstab OBJECT-TYPE
        SYNTAX  Gauge
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcp 9 }

tcpInSegs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcp 10 }

tcpOutSegs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcp 11 }

tcpRetransSegs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcp 12 }

-- the TCP connections table

tcpConnTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF TcpConnEntry
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcp 13 }

tcpConnEntry OBJECT-TYPE
        SYNTAX  TcpConnEntry
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcpConnTable 1 }

TcpConnEntry ::= SEQUENCE {
    tcpConnState
        INTEGER,
    tcpConnLocalAddress
```

```
        IpAddress,
    tcpConnLocalPort
        INTEGER (0..65535),
    tcpConnRemAddress
        IpAddress,
    tcpConnRemPort
        INTEGER (0..65535)
}

tcpConnState OBJECT-TYPE
        SYNTAX  INTEGER {
                    closed(1),
                    listen(2),
                    synSent(3),
                    synReceived(4),
                    established(5),
                    finWait1(6),
                    finWait2(7),
                    closeWait(8),
                    lastAck(9),
                    closing(10),
                    timeWait(11)
                }
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcpConnEntry 1 }

tcpConnLocalAddress OBJECT-TYPE
        SYNTAX  IpAddress
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcpConnEntry 2 }

tcpConnLocalPort OBJECT-TYPE
        SYNTAX  INTEGER (0..65535)
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcpConnEntry 3 }

tcpConnRemAddress OBJECT-TYPE
        SYNTAX  IpAddress
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcpConnEntry 4 }

tcpConnRemPort OBJECT-TYPE
        SYNTAX  INTEGER (0..65535)
        ACCESS  read-only
```

```
        STATUS  mandatory
        ::= { tcpConnEntry 5 }

-- the UDP group

udpInDatagrams OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { udp 1 }

udpNoPorts OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { udp 2 }

udpInErrors OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { udp 3 }

udpOutDatagrams OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { udp 4 }

-- the EGP group

egpInMsgs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { egp 1 }

egpInErrors OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { egp 2 }

egpOutMsgs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { egp 3 }
```

```
egpOutErrors OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { egp 4 }

-- the EGP Neighbor table

egpNeighTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF EgpNeighEntry
        ACCESS  read-only
        STATUS  mandatory
        ::= { egp 5 }

egpNeighEntry OBJECT-TYPE
        SYNTAX  EgpNeighEntry
        ACCESS  read-only
        STATUS  mandatory
        ::= { egpNeighTable 1 }

EgpNeighEntry ::= SEQUENCE {
    egpNeighState
        INTEGER,
    egpNeighAddr
        IpAddress
}

egpNeighState OBJECT-TYPE
        SYNTAX  INTEGER {
                    idle(1),
                    acquisition(2),
                    down(3),
                    up(4),
                    cease(5)
                }
        ACCESS  read-only
        STATUS  mandatory
        ::= { egpNeighEntry 1 }

egpNeighAddr OBJECT-TYPE
        SYNTAX  IpAddress
        ACCESS  read-only
        STATUS  mandatory
        ::= { egpNeighEntry 2 }

END
```

7. Acknowledgements

The initial draft of this memo was heavily influenced by the the HEMS [9] and SNMP [10] MIBs.

Its final form is the result of the suggestions, the dicussions, and the compromises reached by the members of the IETF MIB working group:

Karl Auerbach, Epilogue Technology
K. Ramesh Babu, Excelan
Lawrence Besaw, Hewlett-Packard
Jeffrey D. Case, University of Tennessee at Knoxville
James R. Davin, Proteon
Mark S. Fedor, NYSERNet
Robb Foster, BBN
Phill Gross, The MITRE Corporation
Bent Torp Jensen, Convergent Technology
Lee Labarre, The MITRE Corporation
Dan Lynch, Advanced Computing Environments
Keith McCloghrie, The Wollongong Group
Dave Mackie, 3Com/Bridge
Craig Partridge, BBN (chair)
Jim Robertson, 3Com/Bridge
Marshall T. Rose, The Wollongong Group
Greg Satz, cisco
Martin Lee Schoffstall, Rensselaer Polytechnic Institute
Lou Steinberg, IBM
Dean Throop, Data General
Unni Warrier, Unisys

8. References

[1] Cerf, V., "IAB Recommendations for the Development of Internet Network Management Standards", RFC 1052, IAB, April 1988.

[2] Information processing systems - Open Systems Interconnection, "Management Information Services Definition", International Organization for Standardization, Draft Proposal 9595/2, December 1987.

[3] Information processing systems - Open Systems Interconnection, "Management Information Protocol Specification", International Organization for Standardization, Draft Proposal 9596/2, December 1987.

[4] Rose M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based internets", RFC 1065, TWG, August 1988.

[5] Partridge C., and G. Trewitt, "The High-Level Entity Management System (HEMS)", RFCs 1021-1024, BBN and Stanford, October 1987.

[6] Cerf, V., "Report of the Second Ad Hoc Network Management Review Group", RFC 1109, IAB, August 1989.

[7] Rose, M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based Internets", RFC 1155, Performance Systems International and Hughes LAN Systems, May 1990.

[8] Case, J., M. Fedor, M. Schoffstall, and J. Davin, The Simple Network Management Protocol", RFC 1157, University of Tennessee at Knoxville, Performance Systems International, Performance Systems International, and the MIT Laboratory for Computer Science, May 1990.

[9] Partridge C., and G. Trewitt, "HEMS Variable Definitions", RFC 1024, BBN and Stanford, October 1987.

[10] Case, J., M. Fedor, M. Schoffstall, and J. Davin, "A Simple Network Management Protocol", RFC 1067, University of Tennessee At Knoxville, NYSERNet, Rensselaer Polytechnic, Proteon, August 1988.

[11] LaBarre, L., "Structure and Identification of Management Information for the Internet", Internet Engineering Task Force working note, Network Information Center, SRI International, Menlo Park, California, April 1988.

[12] LaBarre, L., "Transport Layer Management Information: TCP", Internet Engineering Task Force working note in preparation. Network Information Center, SRI International, Menlo Park, California, (unpublished).

[13] Information processing systems - Open Systems Interconnection, "Specification of Abstract Syntax Notation One (ASN.1)", International Organization for Standardization, International Standard 8824, December 1987.

[14] Information processing systems - Open Systems Interconnection, "Specification of Basic Encoding Rules for Abstract Notation One (ASN.1)", International Organization for Standardization, International Standard 8825, December 1987.

[15] Jacobson, V., "Congestion Avoidance and Control", SIGCOMM, 1988, Stanford, California.

Security Considerations

Security issues are not discussed in this memo.

Authors' Addresses

Keith McCloghrie
The Wollongong Group
1129 San Antonio Road
Palo Alto, CA 04303

Phone: (415) 962-7160

EMail: sytek!kzm@HPLABS.HP.COM

Marshall T. Rose
PSI, Inc.
PSI California Office
P.O. Box 391776
Mountain View, CA 94039

Phone: (415) 961-3380

EMail: mrose@PSI.COM

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	)	Art Unit:
Lakshmi Arunachalam	)	Examiner:
Serial No. 11/980,185	)	
Filing Date: Oct. 30, 2008	)	
Title: METHOD AND APPARAUTUS FOR ENABLING REAL TIME TRANSACTIONS ON A NETWORK	)	

INFORMATION DISCLOSURE STATEMENT

Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

Honorable Commissioner:

In accordance with 37 C.F.R. §1.97, please accept this Information Disclosure Statement and copies of any non-US patent art.

COMMENTS

It is believed that this disclosure complies with 37 C.F.R. §1.56 and 1.98 and M.P.E.P. §2000. This disclosure statement should not be construed as a representation that a search has been made or that no other material information as defined in 37 C.F.R. §1.56(a) exists. A copy of each non-US patent reference is being

Respectfully Submitted

Clifford Kraft

Clifford H. Kraft
Reg. No. 35,229
Attorney of Record

CORRESPONDENCE ADDRESS

Clifford H. Kraft
320 Robin Hill Dr.
Naperville, IL 60540

708 528-9092 Tel.
630 393-9114 Fax.

First Class Mailing Certificate: This paper is being filed by United States First Class Mail addressed to Commissioner for Patents, P.O. Box 1450, Alexandria VA. 22313-1450 with sufficient postage on:

Date: Nov. 18, 2008

Signature: Clifford Kraft

Name: Clifford H. Kraft

PTO/SB/08a (10-08)
Approved for use through 11/30/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	11/980,185
Filing Date	10-30-2008
First Named Inventor	Arunachalam
Art Unit	
Examiner Name	
Attorney Docket Number	

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] *(if known)*	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]-Number[4]-Kind Code[5] *(if known)*	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]
		WO 97/18515	05-22-1997	Arunachalam		

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional). [2] See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 13/00

A1

(11) International Publication Number: WO 97/18515

(43) International Publication Date: 22 May 1997 (22.05.97)

(21) International Application Number: PCT/US96/18165

(22) International Filing Date: 13 November 1996 (13.11.96)

(30) Priority Data:

60/006,634	13 November 1995 (13.11.95)	US
08/700,726	5 August 1996 (05.08.96)	US

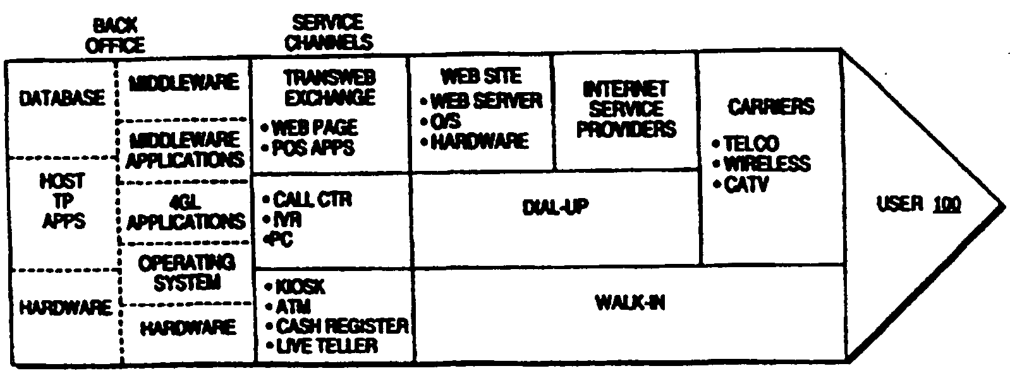
(71)(72) Applicant and Inventor: ARUNACHALAM, Lakshmi [US/US]; 222 Stanford Avenue, Menlo Park, CA 94025 (US).

(74) Agents: SALTER, James, H. et al.; Blakely, Sokoloff, Taylor & Zafman, 7th floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025-1026 (US).

(81) Designated States: AL, AM, AT, AT (Utility model), AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, CZ (Utility model), DE, DE (Utility model), DK, DK (Utility model), EE, EE (Utility model), ES, FI, FI (Utility model), GB, GE, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SK (Utility model), TJ, TM, TR, TT, UA, UG, US, UZ, VN, ARIPO patent (KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: A METHOD AND APPARATUS FOR CONFIGURABLE VALUE-ADDED NETWORK (VAN) SWITCHING AND OBJECT ROUTING



(57) Abstract

The present invention provides a method (802-818) and apparatus (200, 205, 206, 208) for providing real-time, two-way transactional capabilities on the Web. Specifically, one embodiment of the present invention discloses a configurable value-added network switch (520) for enabling real-time transactions on the World Wide Web. The configurable value added network switch (520) comprises a device (812) for switching to a transactional application in response to a user specification (100) from a World Wide Web application, a device (812) for transmitting a transaction request from the transactional application, and a device (814) for processing the transaction request. Additionally, a method for enabling object routing is disclosed, comprising the steps of creating a virtual information store containing information entries and attributes associating each of the information entries and the attributes with an object identity, and assigning a unique network address to each of the object identities. Finally, a method is disclosed for enabling service management of the value-added network service, to perfrom OMA & P functions on the services network.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia	GB	United Kingdom	MW	Malawi
AT	Austria	GE	Georgia	MX	Mexico
AU	Australia	GN	Guinea	NE	Niger
BB	Barbados	GR	Greece	NL	Netherlands
BE	Belgium	HU	Hungary	NO	Norway
BF	Burkina Faso	IE	Ireland	NZ	New Zealand
BG	Bulgaria	IT	Italy	PL	Poland
BJ	Benin	JP	Japan	PT	Portugal
BR	Brazil	KE	Kenya	RO	Romania
BY	Belarus	KG	Kyrgystan	RU	Russian Federation
CA	Canada	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic			SE	Sweden
CG	Congo	KR	Republic of Korea	SG	Singapore
CH	Switzerland	KZ	Kazakhstan	SI	Slovenia
CI	Côte d'Ivoire	LI	Liechtenstein	SK	Slovakia
CM	Cameroon	LK	Sri Lanka	SN	Senegal
CN	China	LR	Liberia	SZ	Swaziland
CS	Czechoslovakia	LT	Lithuania	TD	Chad
CZ	Czech Republic	LU	Luxembourg	TG	Togo
DE	Germany	LV	Latvia	TJ	Tajikistan
DK	Denmark	MC	Monaco	TT	Trinidad and Tobago
EE	Estonia	MD	Republic of Moldova	UA	Ukraine
ES	Spain	MG	Madagascar	UG	Uganda
FI	Finland	ML	Mali	US	United States of America
FR	France	MN	Mongolia	UZ	Uzbekistan
GA	Gabon	MR	Mauritania	VN	Viet Nam

A METHOD AND APPARATUS FOR CONFIGURABLE VALUE-ADDED NETWORK (VAN) SWITCHING AND OBJECT ROUTING

RELATED APPLICATIONS

This application claims the benefit under Title 35, United States Code, Section 119(e) of the United States provisional application having the serial number 60/006,634, filed on November 13th, 1995.

FIELD OF THE INVENTION

The present invention relates to the area of Internet communications. Specifically, the present invention relates to a method and apparatus for configurable value-added network switching and object routing.

BACKGROUND OF THE INVENTION

With the Internet and the World Wide Web ("the Web") evolving rapidly as a viable consumer medium for electronic commerce, new on-line services are emerging to fill the needs of on-line users. An Internet user today can browse on the Web via the use of a Web browser. Web browsers are software interfaces that run on Web clients to allow access to Web servers via a simple user interface. A Web user's capabilities today from a Web browser are, however, extremely limited. The user can perform one-way, browse-only interactions. Additionally, the user has limited "deferred" transactional capabilities, namely electronic mail (e-mail) capabilities. E-mail capabilities are referred to as "deferred transactions" because the consumer's request is not processed until the e-mail is received, read, and the person or system reading the e-mail executes the transaction. This transaction is thus not performed in real-time.

Figure 1A illustrates typical user interactions on the Web today. User 100 sends out a request from Web browser 102 in the form of a universal resource locator (URL) 101 in the following manner:

http://www.car.com. URL 101 is processed by Web browser 102 that determines the URL corresponds to car dealer Web page 105, on car dealer Web server 104. Web browser 102 then establishes browse link 103 to car dealer Web page 105. User 100 can browse Web page 105 and select "hot links" to jump to other locations in Web page 105, or to move to other Web pages on the Web. This interaction is typically a browse-only interaction. Under limited circumstances, the user may be able to fill out a form on car dealer Web page 105, and e-mail the form to car dealer Web server 104. This interaction is still strictly a one-way browse mode communications link, with the e-mail providing limited, deferred transactional capabilities.

Under limited circumstances, a user may have access to two-way services on the Web via Common Gateway Interface (CGI) applications. CGI is a standard interface for running external programs on a Web server. It allows Web servers to create documents dynamically when the server receives a request from the Web browser. When the Web server receives a request for a document, the Web server dynamically executes the appropriate CGI script and transmits the output of the execution back to the requesting Web browser. This interaction can thus be termed a "two-way" transaction. It is a severely limited transaction, however, because each CGI application is customized for a particular type of application or service.

For example, as illustrated in **Figure 1B**, user 100 may access bank 150's Web server and attempt to perform transactions on checking account 152 and to make a payment on loan account 154. In order for user 100 to access checking account 152 and loan account 154 on the Web, CGI application scripts must be created for each account, as illustrated in **Figure 1B**. The bank thus has to create individual scripts for each of its services to offer users access to these services. User 100 can then interact in a limited fashion with these individual applications. Creating and managing individual CGI scripts for each service is not a viable solution for merchants with a large number of services.

As the Web expands and electronic commerce becomes more desirable, the need increases for robust, real-time, bi-directional transactional capabilities on the Web. A true real-time, bi-directional transaction would allow a user to connect to a variety of services on the Web, and perform real-time transactions on those services. For example, although user 100 can browse car dealer Web page 105 today, the user cannot purchase the car, negotiate a car loan or perform other types of real-time, two-way transactions that he can perform with a live salesperson at the car dealership. Ideally, user 100 in **Figure 1A** would be able to access car dealer Web page 105, select specific transactions that he desires to perform, such as purchase a car, and perform the purchase in real-time, with two-way interaction capabilities. CGI applications provide user 100 with a limited ability for two-way interaction with car dealer Web page 105, but due to the lack of interaction and management between the car dealer and the bank, he will not be able to obtain a loan and complete the purchase of the car via a CGI application. The ability to complete robust real-time, two-way transactions is thus not truly available on the Web today.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a method and apparatus for providing real-time, two-way transactional capabilities on the Web. Specifically, one embodiment of the present invention discloses a configurable value-added network switch for enabling real-time transactions on the World Wide Web. The configurable value added network switch comprises means for switching to a transactional application in response to a user specification from a World Wide Web application, means for transmitting a transaction request from the transactional application, and means for processing the transaction request.

According to another aspect of the present invention, a method and apparatus for enabling object routing on the World Wide Web is disclosed. The method for enabling object routing comprises the steps of creating a virtual information store containing information entries and attributes, associating each of the information entries and the attributes with an object identity, and assigning a unique network address to each of the object identities.

Other objects, features and advantages of the present invention will be apparent from the accompanying drawings and from the detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention will be apparent from the accompanying drawings and from the detailed description of the present invention as set forth below.

Figure 1A is an illustration of a current user's browse capabilities on the Web via a Web browser.

Figure 1B is an illustration of a current user's capabilities to perform limited transactions on the Web via CGI applications.

Figure 2 illustrates a typical computer system on which the present invention may be utilized.

Figure 3 illustrates the Open Systems Interconnection (OSI) Model.

Figure 4A illustrates conceptually the user value chain as it exists today.

Figure 4B illustrates one embodiment of the present invention.

Figure 5A illustrates a user accessing a Web server including one embodiment of the present invention.

Figure 5B illustrates the exchange component according to one embodiment of the present invention.

Figure 5C illustrates an example of a point-of-service (POSvc) application list.

Figure 5D illustrates a user selecting a bank POSvc application from the POSvc application list.

Figure 5E illustrates a three-way transaction according to one embodiment of the present invention.

Figure 6A illustrates a value-added network (VAN) switch.

Figure 6B illustrates the hierarchical addressing tree structure of the networked objects in DOLSIBs.

Figure 7 illustrates conceptually the layered architecture of a VAN switch.

Figure 8 is a flow diagram illustrating one embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention relates to a method and apparatus for configurable value-added network switching and object routing and management. "Web browser" as used in the context of the present specification includes conventional Web browsers such as NCSA Mosaic™ from NCSA and Netscape Mosaic™ from Netscape™. The present invention is independent of the Web browser being utilized and the user can use any Web browser, without modifications to the Web browser. In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent to one of ordinary skill in the art, however, that these specific details need not be used to practice the present invention. In other instances, well-known structures, interfaces and processes have not been shown in detail in order not to unnecessarily obscure the present invention.

Figure 2 illustrates a typical computer system 200 in which the present invention operates. The preferred embodiment of the present invention is implemented on an IBM™ Personal Computer manufactured by IBM Corporation of Armonk, New York. Alternate embodiments may be implemented on a Macintosh™ computer manufactured by Apple™ Computer, Incorporated of Cupertino, California. It will be apparent to those of ordinary skill in the art that other alternative computer system architectures may also be employed.

In general, such computer systems as illustrated by **Figure 2** comprise a bus 201 for communicating information, a processor 202 coupled with the bus 201 for processing information, main memory 203 coupled with the bus 201 for storing information and instructions for the processor 202, a read-only memory 204 coupled with the bus 201 for storing static information and instructions for the processor 202, a display device 205 coupled with the bus 201 for displaying information for a computer user, an input device 206 coupled with the bus 201 for communicating information and command selections to the processor 202, and a mass storage device 207, such as a magnetic disk and associated disk drive, coupled with the bus 201 for storing information and instructions. A data storage medium 208 containing digital information is configured to operate with mass storage device 207 to allow processor 202 access to the digital information on data storage medium 208 via bus 201.

Processor 202 may be any of a wide variety of general purpose processors or microprocessors such as the Pentium™ microprocessor manufactured by Intel™ Corporation or the Motorola™ 68040 or Power PC™ brand microprocessor manufactured by manufactured by Motorola™ Corporation. It will be apparent to those of ordinary skill in the art, however, that other varieties of processors may also be used in a particular computer system. Display device 205 may be a liquid crystal device, cathode ray tube (CRT), or other suitable display device. Mass storage device 207 may be a conventional hard disk drive, floppy disk drive, CD-ROM drive, or other magnetic or optical data storage device for reading and writing information stored on a hard disk, a floppy disk, a CD-ROM a magnetic tape, or other magnetic or optical data storage medium. Data storage medium 208 may be a hard disk, a floppy disk, a CD-ROM, a magnetic tape, or other magnetic or optical data storage medium.

In general, processor 202 retrieves processing instructions and data from a data storage medium 208 using mass storage device 207 and downloads this information into random access memory 203 for execution. Processor 202, then executes an instruction stream from random access memory 203 or read-only memory 204. Command selections and information input at input device 206 are used to direct the flow of instructions executed by processor 202. Equivalent input device 206 may also be a pointing device such as a conventional mouse or trackball device. The results of this processing execution are then displayed on display device 205.

The preferred embodiment of the present invention is implemented as a software module, which may be executed on a computer system such as computer system 200 in a conventional manner. Using well known techniques, the application software of the preferred embodiment is stored on data storage medium 208 and subsequently loaded into and executed within computer system 200. Once initiated, the software of the preferred embodiment operates in the manner described below.

Figure 3 illustrates the Open Systems Interconnection (OSI) reference model. OSI Model 300 is an international standard that provides a common basis for the coordination of standards development, for the purpose of systems interconnection. The present invention is implemented to function as a routing switch within the "application layer" of the OSI model. The model defines seven layers, with each layer communicating with its peer layer in another node through the use of a protocol. Physical layer 301 is the lowest layer, with responsibility to transmit unstructured bits across a link. Data link layer 302 is the next layer above physical layer 301. Data link layer 302 transmits chunks across the link and deals with problems like checksumming to detect data corruption, orderly coordination of the use of shared media and addressing when multiple systems are reachable. Network bridges operate within data link layer 302.

Network layer 303 enables any pair of systems in the network to communicate with each other. Network layer 303 contains hardware units such as routers, that handle routing, packet fragmentation and reassembly of packets. Transport layer 304 establishes a reliable communication stream between a pair of systems, dealing with errors such as lost packets, duplicate packets, packet reordering and fragmentation. Session layer 305 offers services above the simple communication stream provided by transport layer 304. These services include dialog control and chaining. Presentation layer 306 provides a means by which OSI compliant applications can agree on representations for data. Finally, application layer 307 includes services such as file transfer, access and management services (FTAM), electronic mail and virtual terminal (VT) services. Application layer 307 provides a means for application programs to access the OSI environment. As described above, the present invention is implemented to function as a routing switch in application layer 307. Application layer routing creates an open channel for the management, and the selective flow of data from remote databases on a network.

A. OVERVIEW

Figure 4A illustrates conceptually the user value chain as it exists today. The user value chain in **Figure 4A** depicts the types of transactions that are performed today, and the channels through which the transactions are performed. A "transaction" for the purposes of the present invention includes any type of commercial or other type of interaction that a user may want to perform. Examples of transactions include a deposit into a bank account, a request for a loan from a bank, a purchase of a car from a car dealership or a purchase of a car with financing from a bank. A large variety of other transactions are also possible.

A typical user transaction today may involve user 100 walking into a bank or driving up to a teller machine, and interacting with a live bank teller, or automated teller machine (ATM) software applications. Alternatively, user 100 can perform the same transaction by using a personal computer (PC), activating application software on his PC to access his bank account, and dialing into the bank via a modem line. If user 100 is a Web user, however, there is no current mechanism for performing a robust, real-time transaction with the bank, as illustrated in **Figure 4A**. CGI scripts provide only limited two-way capabilities, as described above. Thus, due to this lack of a robust mechanism by which real-time Web transactions can be performed, the bank is unable to be a true "Web merchant," namely a merchant capable of providing complete transactional services on the Web.

According to one embodiment of the present invention, as illustrated in **Figure 4B**, each merchant that desires to be a Web merchant can provide real-time transactional capabilities to users who desire to access the merchants' services via the Web. This embodiment includes a service network running on top of a facilities network, namely the Internet, the Web or e-mail networks. For the purposes of this application, users are described as utilizing PC's to access the Web via Web server "switching" sites. (Switching is described in more detail below). Users may also utilize other personal devices such as network computers or cellular devices to access the merchants' services via appropriate switching sites. These switching sites include non-Web network computer sites and cellular provider sites. Five components interact to provide this service network functionality, namely an exchange, an operator agent, a management agent, a management manager and a graphical user interface. All five components are described in more detail below.

As illustrated in **Figure 5A**, user 100 accesses Web server 104. Having accessed Web server 104, user 100 can decide that he desires to perform real-time transactions. When Web server 104 receives user 100's indication that he desires to perform real-time transactions, the request is handed over to an exchange component. Thus, from Web page 105, for example, user 100 can select button 500, entitled "Transactions" and Web server 104 hands user 100's request over to the exchange component. The button and the title can be replaced by any mechanism that can instruct a Web server to hand over the consumer's request to the exchange component.

Figure 5B illustrates exchange 501. Exchange 501 comprises Web page 505 and point-of-service (POSvc) applications 510. Exchange 501 also conceptually includes a switching component and an object routing component (described in more detail below). POSvc applications 510 are transactional applications, namely applications that are designed to incorporate and take advantage of the capabilities provided by the present invention. Although exchange 501 is depicted as residing on Web server 104, the exchange can also reside on a separate computer system that resides on the Internet and has an Internet address. Exchange 501 may also include operator agent 503 that interacts with a management manager (described in more detail below). Exchange 501 creates and allows for the management (or distributed control) of a service network, operating within the boundaries of an IP-based facilities network. Thus, exchange 501 and a management agent component, described in more detail below, under the headings "VAN Switch and Object Routing," together perform the switching, object routing, application and service management functions according to one embodiment of the present invention.

Exchange 501 processes the consumer's request and displays an exchange Web page 505 that includes a list of POSvc applications 510 accessible by exchange 501. A POSvc application is an application that can execute the type of transaction that the user may be interested in performing. The POSvc list is displayed via the graphical user interface component. One embodiment of the present invention supports HyperText Markup Language as the graphical user interface component. Virtual Reality Markup Language and Java™ are also supported by this embodiment. A variety of other graphical user interface standards can also be utilized to implement the graphical user interface.

An example of a POSvc application list is illustrated in **Figure 5C**. User 100 can thus select from POSvc applications Bank 510(1), Car Dealer 510(2) or Pizzeria 510(3). Numerous other POSvc applications can also be included in this selection. If user 100 desires to perform a number of banking transactions, and selects the Bank application, a Bank POSvc application will be activated and presented to user 100, as illustrated in **Figure 5D**. For the purposes of illustration, exchange 501 in **Figure 5D** is shown as running on a different computer system (Web server 104) from the computer systems of the Web merchants running POSvc applications (computer system 200). Exchange 501 may, however, also be on the same computer system as one or more of the computer systems of the Web merchants.

Once Bank POSvc application 510 has been activated, user 100 will be able to connect to Bank services and utilize the application to perform banking transactions, thus accessing data from a host or data repository 575 in the Bank "Back Office." The Bank Back Office comprises legacy databases and other data repositories that are utilized by the Bank to store its data. This connection between user 100 and Bank services is managed by exchange 501. As illustrated in **Figure 5D**, once the connection is made between Bank POSvc application 510(1), for example, and Bank services, an operator agent on Web server 104 may be activated to ensure the availability of distributed functions and capabilities.

Each Web merchant may choose the types of services that it would like to offer its clients. In this example, if Bank decided to include in their POSvc application access to checking and savings accounts, user 100 will be able to perform real-time transactions against his checking and savings accounts. Thus, if user 100 moves $500 from his checking account into his savings account, the transaction will be performed in real-time, in the same manner the transaction would have been performed by a live teller at the bank or an ATM machine. Therefore, unlike his prior access to his account, user 100 now has the capability to do more than browse his bank account. The ability to perform these types of robust, real-time transactions from a Web client is a significant aspect of the present invention.

Bank can also decide to provide other types of services in POSvc application 510(1). For example, Bank may agree with Car dealership to allow Bank customers to purchase a car from that dealer, request a car loan from Bank, and have the entire transaction performed on the Web, as illustrated in **Figure 5E**. In this instance, the transactions are not merely two-way, between the user and Bank, but three-way, amongst the consumer, Bank and Car dealership. According to one aspect of the present invention, this three-way transaction can be expanded to n-way transactions, where n represents a predetermined number of merchants or other service providers who have agreed to cooperate to provide services to users. The present invention therefore allows for "any-to-any" communication and transactions on the Web, thus facilitating a large, flexible variety of robust, real-time transactions on the Web.

Finally, Bank may also decide to provide intra-merchant or intra-bank services, together with the inter-merchant services described above. For example, if Bank creates a POSvc application for use by the Bank Payroll department, Bank may provide its own employees with a means for submitting timecards for payroll processing by the Bank's Human Resources (HR) Department. An employee selects the Bank HR POSvc application, and submits his timecard. The employee's timecard is processed by accessing the employee's payroll information, stored in the Bank's Back Office. The transaction is thus processed in real-time, and the employee receives his paycheck immediately.

B. VAN SWITCHING AND OBJECT ROUTING

As described above, exchange 501 and management agent 601, illustrated in **Figure 6A**, together constitute a value-added network (VAN) switch. These two elements may take on different roles as necessary, including peer-to-peer, client-server or master-slave roles. Management manager 603 is illustrated as residing on a separate computer system on the Internet. Management manager 603 can, however, also reside on the same machine as exchange 501. Management manager 603 interacts with the operator agent 503 residing on exchange 501.

VAN switch 520 provides multi-protocol object routing, depending upon the specific VAN services chosen. This multi-protocol object routing is provided via a proprietary protocol, TransWeb™ Management Protocol (TMP). TMP incorporates the same security features as the traditional Simple Network Management Protocol, SNMP. It also allows for the integration of other traditional security mechanisms, including RSA security mechanisms.

One embodiment of the present invention utilizes TMP and distributed on-line service information bases (DOLSIBs) to perform object routing. Alternatively, TMP can incorporate s-HTTP, Java™, the WinSock API or ORB with DOLSIBs to perform object routing. DOLSIBs are virtual information stores optimized for networking. All information entries and attributes in a DOLSIB virtual information store are associated with a networked object identity. The networked object identity identifies the information entries and attributes in the DOLSIB as individual networked objects, and each networked object is assigned an Internet address. The Internet address is assigned based on the IP address of the node at which the networked object resides.

For example, in **Figure 5A**, Web server 104 is a node on the Internet, with an IP address. All networked object associated with Web server 104 will therefore be assigned an Internet address based on the Web server 104's IP address. These networked objects thus "branch" from the node, creating a hierarchical tree structure. The Internet address for each networked object in the tree essentially establishes the individual object as an "IP-reachable" or accessible node on the Internet. TMP utilizes this Internet address to uniquely identify and access the object from the DOLSIB. **Figure 6B** illustrates an example of this hierarchical addressing tree structure.

Each object in the DOLSIB has a name, a syntax and an encoding. The name is an administratively assigned object ID specifying an object type. The object type together with the object instance serves to uniquely identify a specific instantiation of the object. For example, if object 610 is information about models of cars, then one instance of that object would provide user 100 with information about a specific model of the car while another instance would provide information about a different model of the car. The syntax of an object type defines the abstract data structure corresponding to that object type. Encoding of objects defines how the object is represented by the object type syntax while being transmitted over the network.

C. MANAGEMENT AND ADMINISTRATION

As described above, exchange 501 and management agent 601 together constitute a VAN switch. **Figure 7** illustrates conceptually the layered architecture of VAN switch 520. Specifically, boundary service 701 provides the interfaces between VAN switch 520, the Internet and the Web, and multi-media end user devices such as PCs, televisions or telephones. Boundary service 701 also provides the interface to the on-line service provider. A user can connect to a local application, namely one accessible via a local VAN switch, or be routed or "switched" to an application accessible via a remote VAN switch.

Switching service 702 is an OSI application layer switch. Switching service 702 thus represents the core of the VAN switch. It performs a number of tasks including the routing of user connections to remote VAN switches, described in the paragraph above, multiplexing and prioritization of requests, and flow control. Switching service 702 also facilitates open systems' connectivity with both the Internet (a public switched network) and private networks including back office networks, such as banking networks. Interconnected application layer switches form the application network backbone. These switches are one significant aspect of the present invention.

Management service 703 contains tools such as Information Management Services (IMS) and application Network Management Services (NMS). These tools are used by the end users to manage network resources, including VAN switches. Management service 703 also provides applications that perform Operations, Administration, Maintenance & Provisioning (OAM&P) functions. These OAM&P functions include security management, fault management, configuration management, performance management and billing management. Providing OAM&P functions for applications in this manner is another significant aspect of the present invention.

Finally, application service 704 contains application programs that deliver customer services. Application service 704 includes POSvc applications such as Bank POSvc described above, and illustrated in **Figure 6A**. Other examples of VAN services include multi-media messaging, archival/retrieval management, directory services, data staging, conferencing, financial services, home banking, risk management and a variety of other vertical services. Each VAN service is designed to meet a particular set of requirements related to performance, reliability, maintenance and ability to handle expected traffic volume. Depending on the type of service, the characteristics of the network elements will differ. VAN service 704 provides a number of functions including communications services for both management and end users of the network and control for the user over the user's environment.

Figure 8 is a flow diagram illustrating one embodiment of the present invention. A user connects to a Web server running an exchange component in step 802. In step 804, the user issues a request for a transactional application, and the web server hands off the request to an exchange in step 806. The exchange activates a graphical user interface to present user with a list of POSvc application options in step 808. In step 810, the user makes a selection from the POSvc application list. In step 812, the switching component in the exchange switches the user to the selected POSvc application, and in step 814, the object routing component executes the user's request. Data is retrieved from the appropriate data repository via TMP in step 816, and finally, the user may optionally continue the transaction in step 818 or end the transaction.

Thus, a configurable value-added network switching and object routing method and apparatus is disclosed. These specific arrangements and methods described herein are merely illustrative of the principles of the present invention. Numerous modifications in form and detail may be made by those of ordinary skill in the art without departing from the scope of the present invention. Although this invention has been shown in relation to a particular preferred embodiment, it should not be considered so limited. Rather, the present invention is limited only by the scope of the appended claims.

CLAIMS

We claim:

1. A configurable value-added network switch for enabling real-time transactions on the World Wide Web, said configurable value added network switch comprising:

means for switching to a transactional application in response to a user specification from a World Wide Web application;

means for transmitting a transaction request from said transactional application; and

means for processing said transaction request.

2. The configurable value-added network switch as claimed in Claim 1 wherein said means for switching further comprises:

means for receiving said user specification;

means for enabling a switch to said transactional application; and

means for activating said transactional application.

3. The configurable value-added network switch as claimed in Claim 2 wherein means for activating said transactional application further includes means for creating a transaction link between said user application and said transactional application.

4. The configurable value-added network switch as claimed in Claim 1 wherein said means for processing said transaction request

further comprises means for coupling said means for transmitting to a host means.

5. The configurable value-added network switch as claimed in Claim 4 wherein said host means contains data corresponding to said transaction request.

6. A method for configuring a value-added network switch on the World Wide Web, said method for configuring said value-added network switch comprising the steps of:

switching to a transactional application in response to a user specification from a World Wide Web application;

transmitting a transaction request from said transactional application; and

processing said transaction request.

7. The method for configuring said value-added network switch as claimed in Claim 6 wherein said means for switching further comprises the steps of:

receiving said user specification;

enabling a switch to said transactional application; and

activating said transactional application.

8. The method for configuring said value-added network switch as claimed in Claim 7 wherein said step of activating said transactional

application further includes the step of creating a transaction link between said user application and said transactional application.

9. The method for configuring said value-added network switch as claimed in Claim 6 wherein said step of processing said transaction request further comprises the step of transmitting said transaction request to a host means.

10. The method for configuring said value-added network switch as claimed in Claim 9 wherein said host means contains data corresponding to said transaction request.

11. A configurable value-added network system for enabling real-time transactions on the World Wide Web, said configurable value-added network system comprising:

means for switching to a transactional application in response to a user specification from a user application;

means for activating an agent to create a transaction link between said user application and said transactional application;

means for transmitting a transaction request from said transactional application; and

a host means for processing said transaction request and retrieving data corresponding to said transaction request.

12. A method for enabling object routing on the World Wide Web, said method for enabling object routing comprising the steps of:

creating a virtual information store containing information entries and attributes;

associating each of said information entries and said attributes with an object identity; and

assigning a unique network address to each of said object identities.

13. The method in claim 12 further comprising the step of utilizing said unique network address to identify and route said object identities on the World Wide Web.

14. The method in claim 12 further comprising the step of utilizing said unique network address to identify and route said object identities on the Internet.

15. The method in claim 12 wherein said step of associating each of said information entries and said attributes with said object identity further includes the step of storing a name, a syntax and an encoding for each of said object identities.

16. The method in claim 15 wherein said name of said object identity specifies an object type.

17. The method in claim 16 wherein said object type and an object instance uniquely identify an instantiation of said object type.

18. The method in claim 17 wherein said syntax defines a data structure for said object type.

19. The method in claim 12 further comprising the step of utilizing said unique network address of each of said object identities to perform Operations, Administration, Maintenance & Provisioning (OAM&P) functions.

20. An object router on the World Wide Web, said object router comprising:

means for creating a virtual information store containing information entries and attributes;

means for associating each of said information entries and said attributes with an object identity; and

means for assigning a unique network address to each of said object identities.

21. The object router in claim 20 further comprising means for utilizing said unique network address to identify and route said object identities on the World Wide Web.

22. The method in claim 20 further comprising means for utilizing said unique network address to identify and route said object identities on the Internet.

23. The method in claim 20 wherein said means for associating each of said information entries and said attributes with said object

identity further includes means for storing a name, a syntax and an encoding for each of said object identities.

24. The method in claim 23 wherein said name of said object identity specifies an object type.

25. The method in claim 24 wherein said object type and an object instance uniquely identify an instantiation of said object type.

26. The method in claim 25 wherein said syntax defines a data structure for said object type.

27. The method in claim 20 further comprising the step of utilizing said unique network address of each of said object identities to perform Operations, Administration, Maintenance & Provisioning (OAM&P) functions.

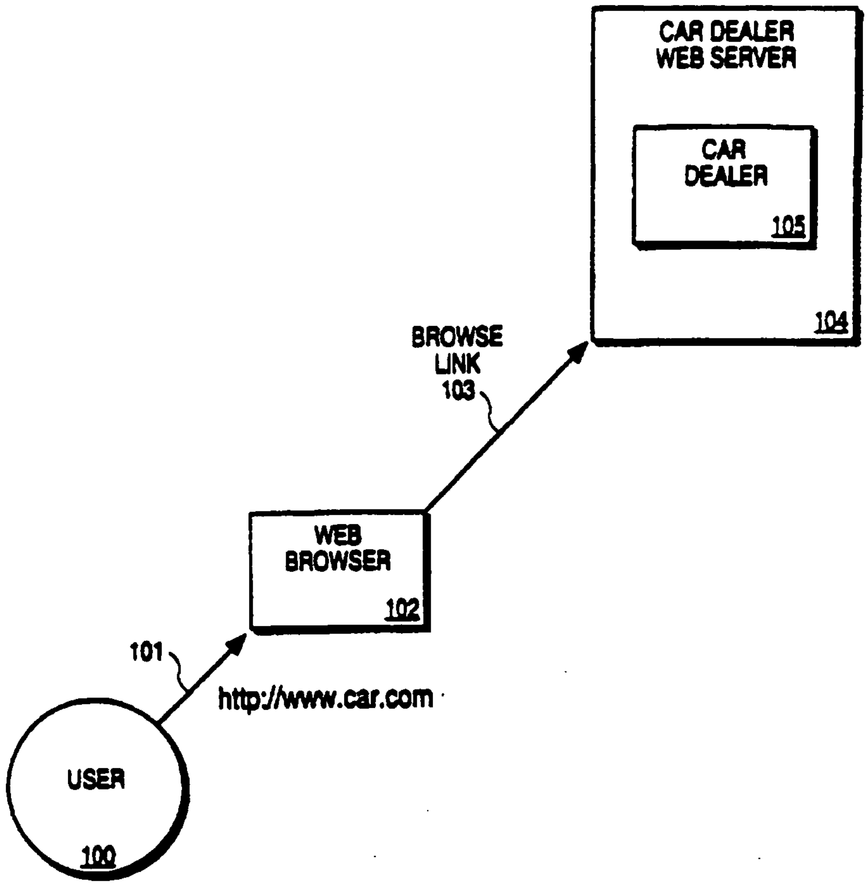


FIG. 1A (PRIOR ART)

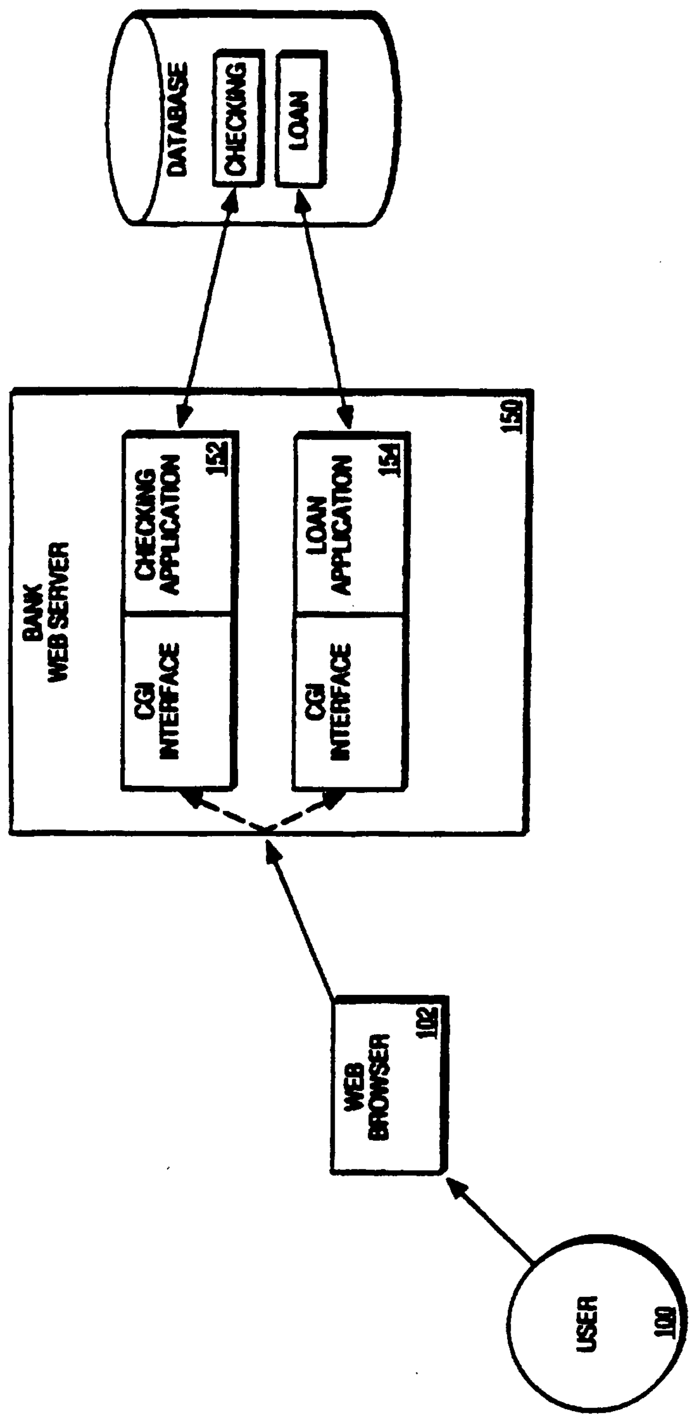


FIG. 1B (PRIOR ART)

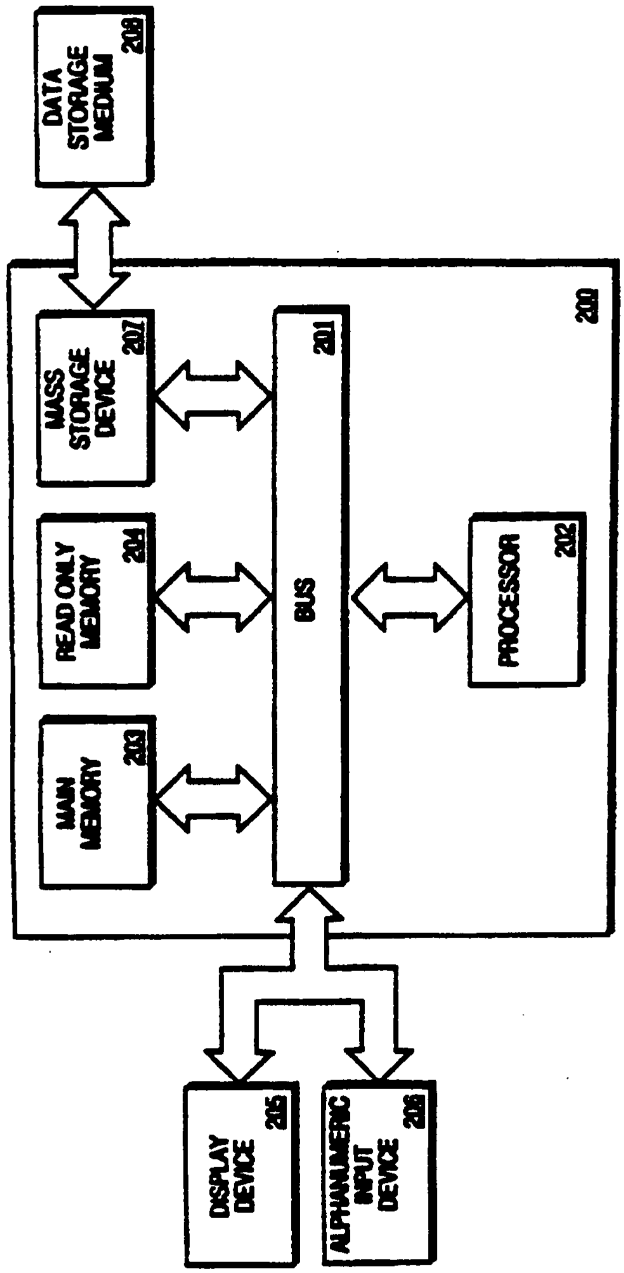

DATA STORAGE MEDIUM
208
MASS STORAGE DEVICE
207
READ ONLY MEMORY
204
MAIN MEMORY
203
BUS
201
PROCESSOR
202
200
DISPLAY DEVICE
205
ALPHANUMERIC INPUT DEVICE
206

FIG. 2

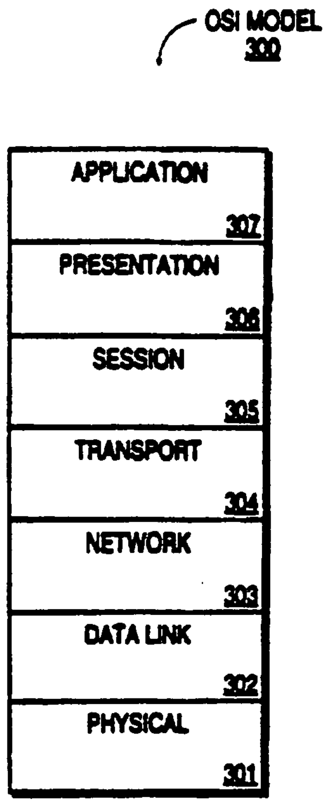

OSI MODEL
300
APPLICATION
307
PRESENTATION
306
SESSION
305
TRANSPORT
304
NETWORK
303
DATA LINK
302
PHYSICAL
301

FIG. 3

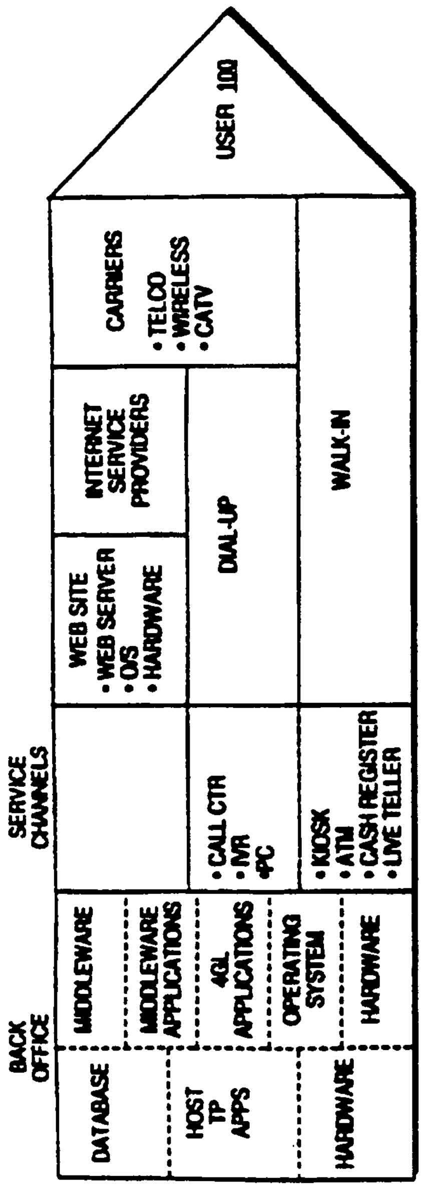

USER 100
CARRIERS
• TELCO
• WIRELESS
• CATV
INTERNET SERVICE PROVIDERS
WEB SITE
• WEB SERVER
• O/S
• HARDWARE
DIAL-UP
WALK-IN
SERVICE CHANNELS
• CALL CTR
• IVR
• PC
• KIOSK
• ATM
• CASH REGISTER
• LIVE TELLER
BACK OFFICE
MIDDLEWARE
MIDDLEWARE APPLICATIONS
4GL APPLICATIONS
OPERATING SYSTEM
HARDWARE
DATABASE
HOST TP APPS
HARDWARE

FIG. 4A

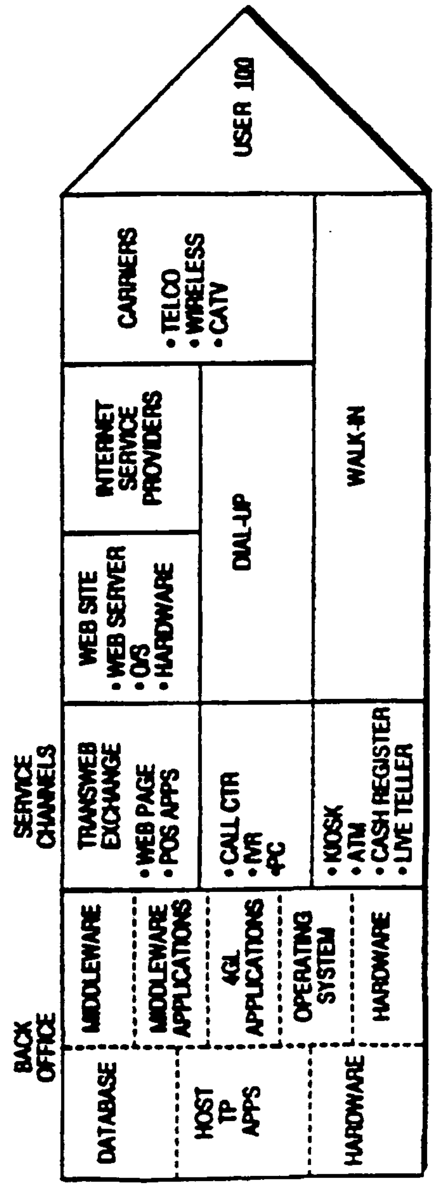

USER 100
CARRIERS
• TELCO
• WIRELESS
• CATV
INTERNET SERVICE PROVIDERS
WEB SITE
• WEB SERVER
• O/S
• HARDWARE
DIAL-UP
WALK-IN
SERVICE CHANNELS
TRANSWEB EXCHANGE
• WEB PAGE
• POS APPS
• CALL CTR
• IVR
• PC
• KIOSK
• ATM
• CASH REGISTER
• LIVE TELLER
BACK OFFICE
MIDDLEWARE
MIDDLEWARE APPLICATIONS
4GL APPLICATIONS
OPERATING SYSTEM
HARDWARE
DATABASE
HOST TP APPS
HARDWARE

FIG. 4B

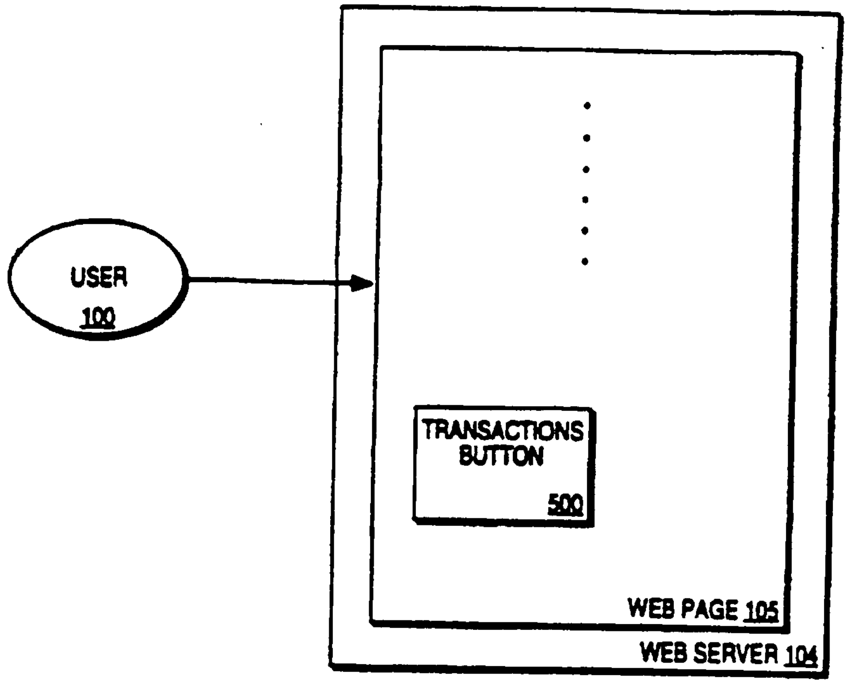

USER
100
TRANSACTIONS BUTTON
500
WEB PAGE 105
WEB SERVER 104

FIG. 5A

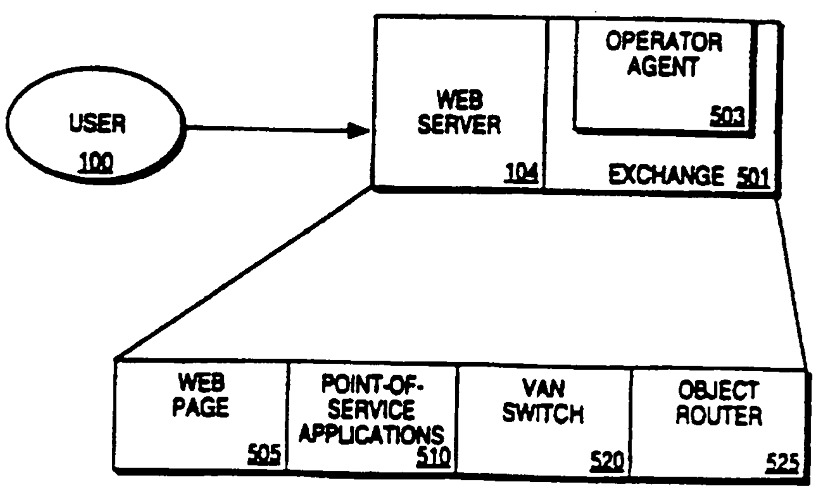

USER
100
WEB SERVER
104
OPERATOR AGENT
503
EXCHANGE 501
WEB PAGE
505
POINT-OF-SERVICE APPLICATIONS
510
VAN SWITCH
520
OBJECT ROUTER
525

FIG. 5B

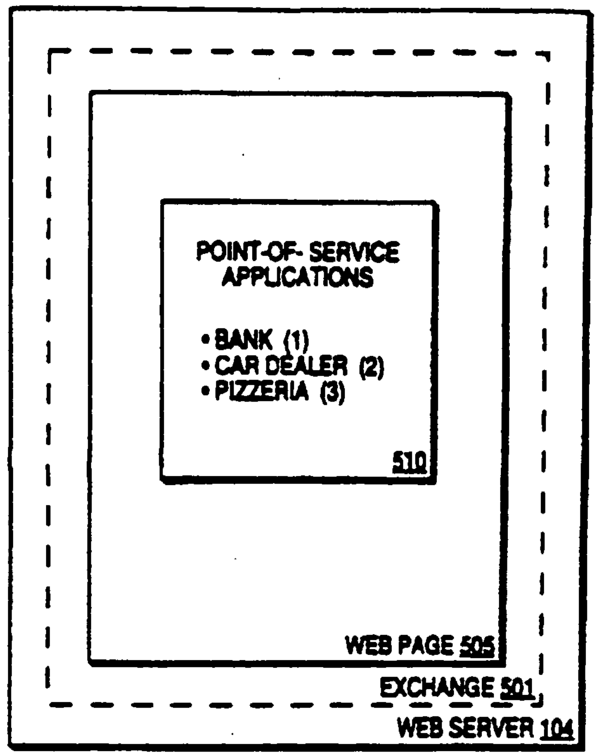

POINT-OF- SERVICE APPLICATIONS
• BANK (1)
• CAR DEALER (2)
• PIZZERIA (3)
510
WEB PAGE 505
EXCHANGE 501
WEB SERVER 104

FIG. 5C

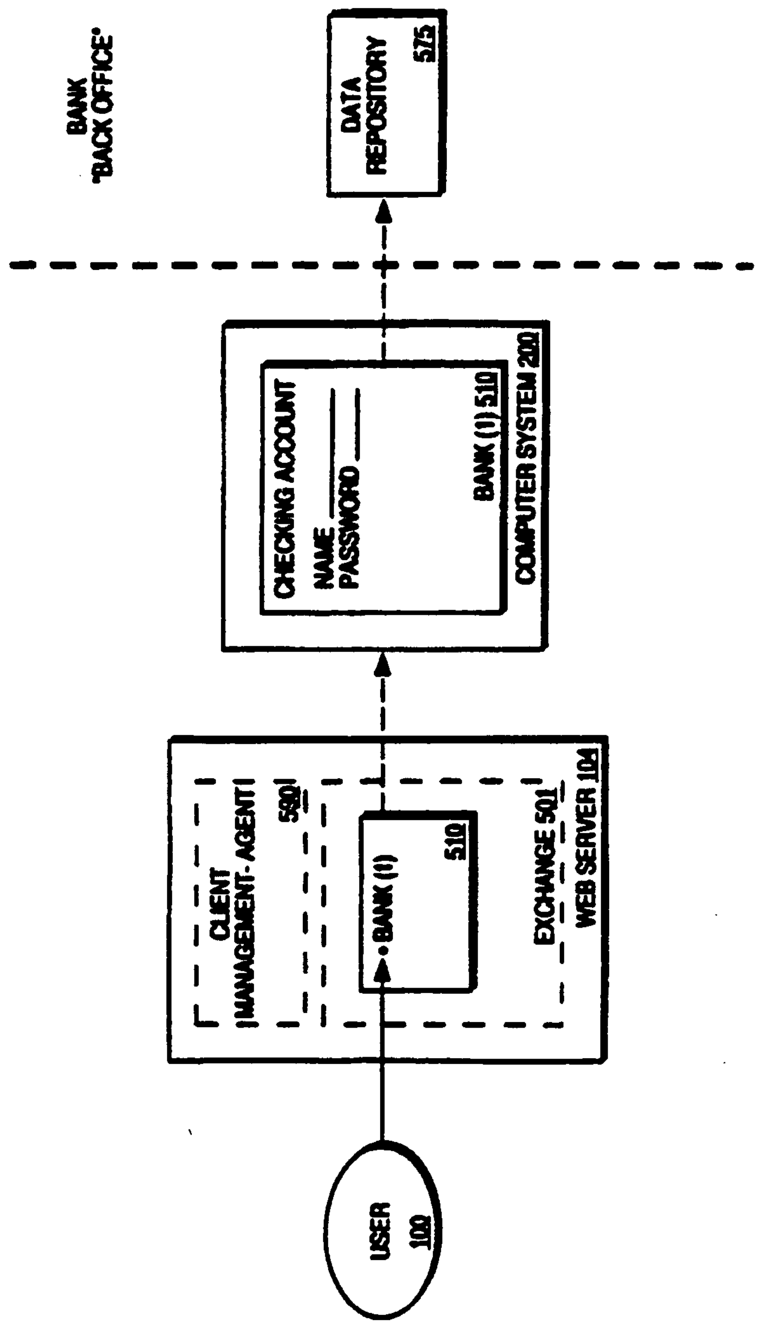

BANK
"BACK OFFICE"
DATA
REPOSITORY
575
CHECKING ACCOUNT
NAME
PASSWORD
BANK (I) 510
COMPUTER SYSTEM 200
CLIENT
MANAGEMENT- AGENT
500
BANK (I)
510
EXCHANGE 501
WEB SERVER 104
USER
100

FIG. 5D

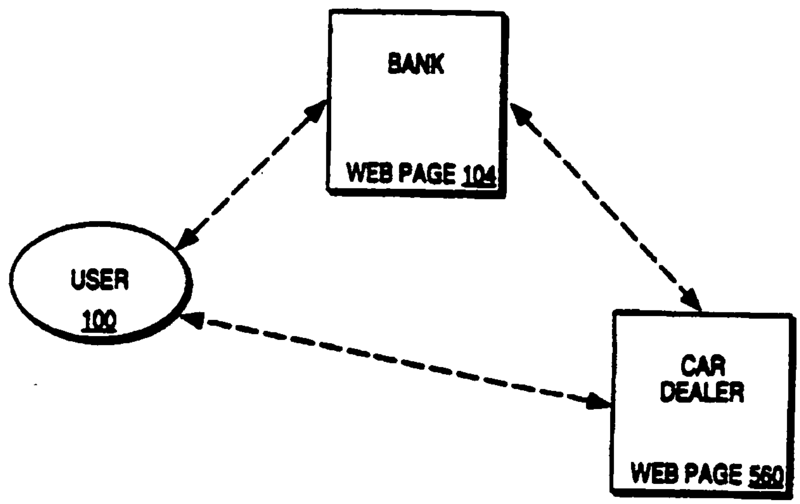


FIG. 5E

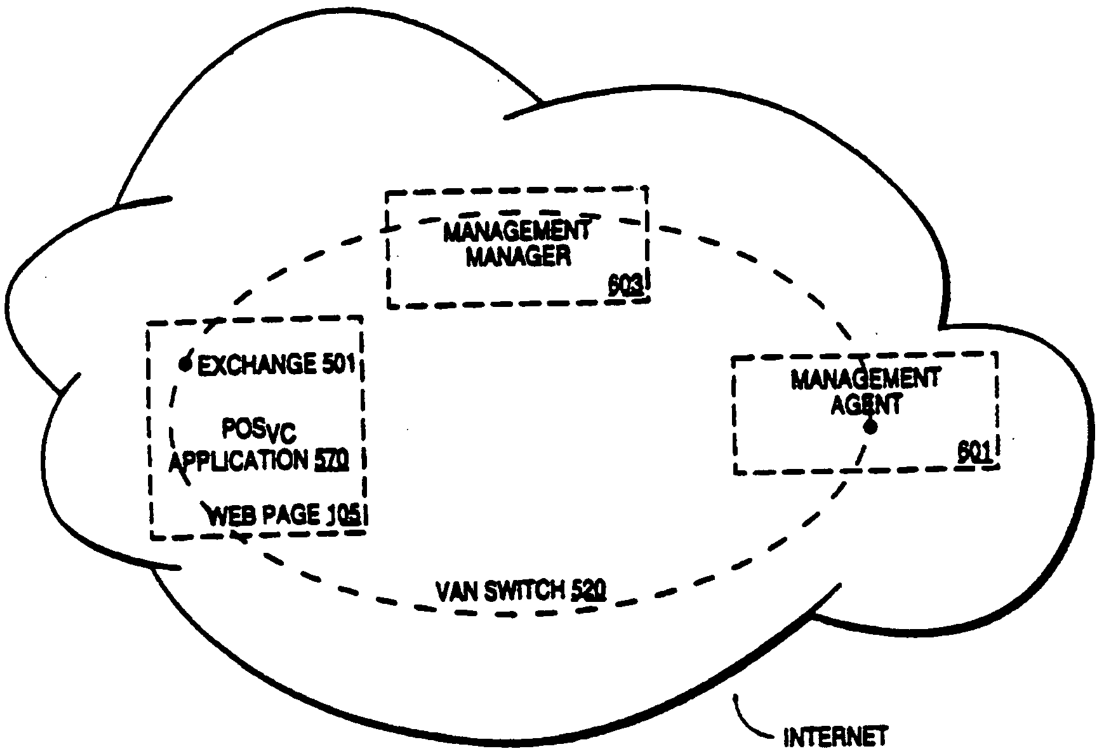

MANAGEMENT
MANAGER
603
EXCHANGE 501
POSVC
APPLICATION 570
WEB PAGE 105
MANAGEMENT
AGENT
601
VAN SWITCH 520
INTERNET

FIG. 6A

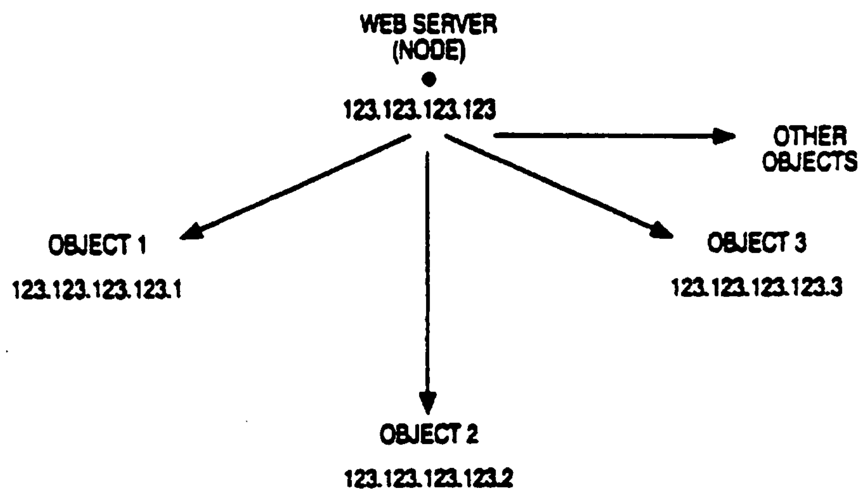

WEB SERVER
(NODE)
123.123.123.123
OTHER
OBJECTS
OBJECT 1
123.123.123.123.1
OBJECT 3
123.123.123.123.3
OBJECT 2
123.123.123.123.2

FIG. 6B

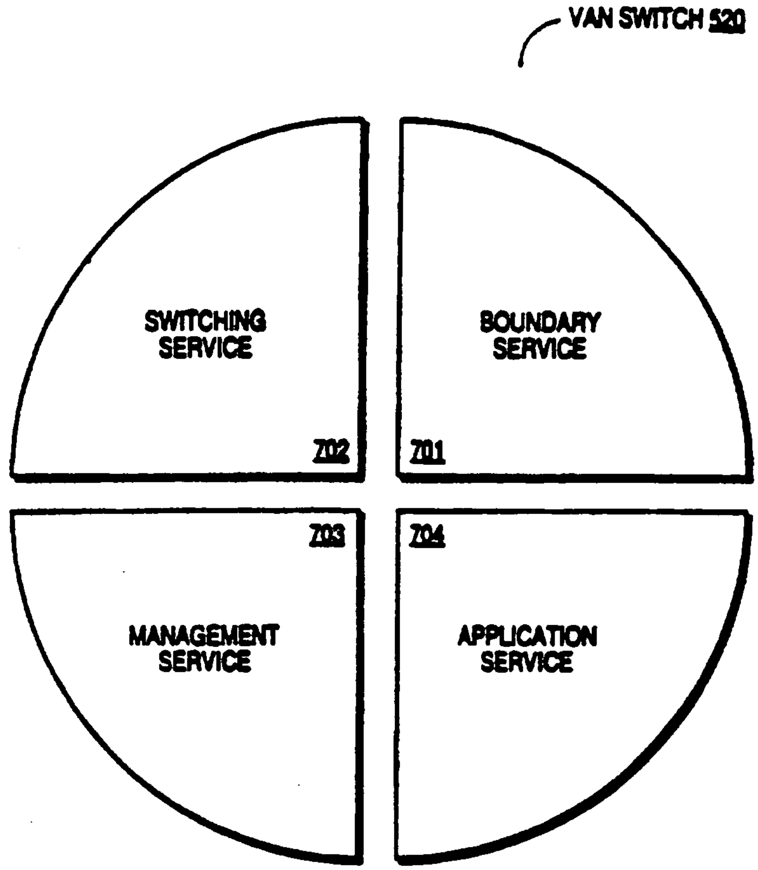

VAN SWITCH 520
SWITCHING SERVICE
702
BOUNDARY SERVICE
701
703
MANAGEMENT SERVICE
704
APPLICATION SERVICE

FIG. 7

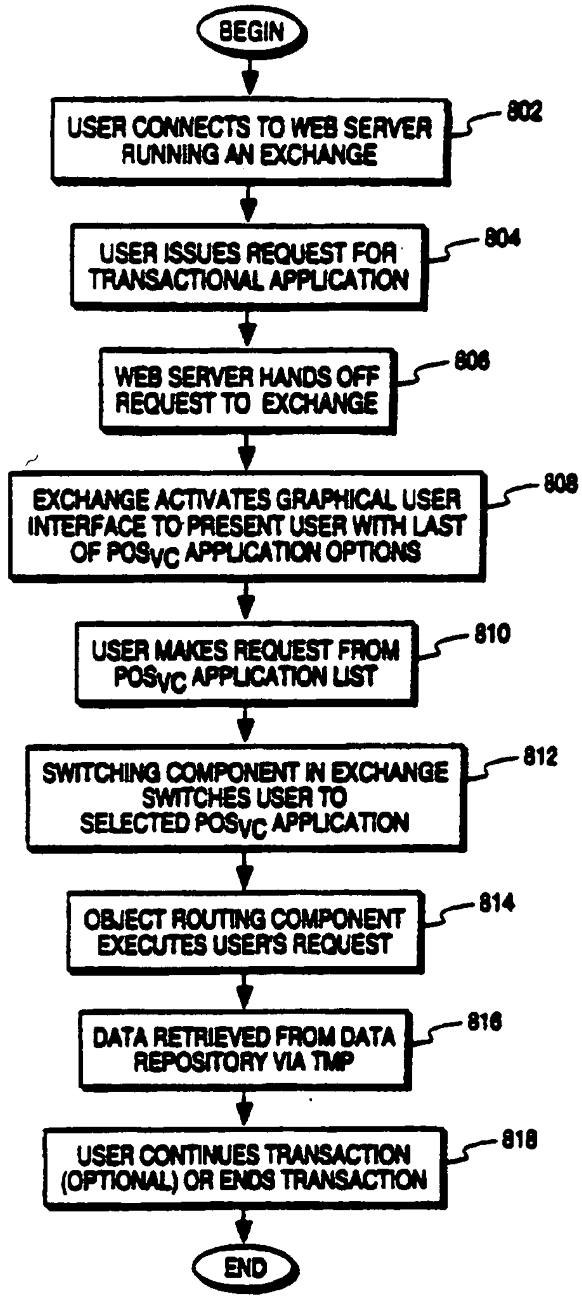

BEGIN
USER CONNECTS TO WEB SERVER RUNNING AN EXCHANGE
802
USER ISSUES REQUEST FOR TRANSACTIONAL APPLICATION
804
WEB SERVER HANDS OFF REQUEST TO EXCHANGE
806
EXCHANGE ACTIVATES GRAPHICAL USER INTERFACE TO PRESENT USER WITH LAST OF POSVC APPLICATION OPTIONS
808
USER MAKES REQUEST FROM POSVC APPLICATION LIST
810
SWITCHING COMPONENT IN EXCHANGE SWITCHES USER TO SELECTED POSVC APPLICATION
812
OBJECT ROUTING COMPONENT EXECUTES USER'S REQUEST
814
DATA RETRIEVED FROM DATA REPOSITORY VIA TMP
816
USER CONTINUES TRANSACTION (OPTIONAL) OR ENDS TRANSACTION
818
END

FIG. 8

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/18165

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) : G06F 13/00
US CL : 395/226
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : US CL : 395/201, 226, 227, 670, 671, 672

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

NONE

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS (USPTO, JPABS), DIALOG

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Dr. Dobb's Journal, Volume 20, No. 6, issued June 1995, Davison, Andrew, "Coding with HTML forms: HTML goes interactive.", pages 70-79, especially page 70.	1-27

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
09 JANUARY 1997	14 FEB 1997
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer ROBERT B. HARRELL Telephone No. (703) 305-9692

Form PCT/ISA/210 (second sheet)(July 1992)★

IKV

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	)	Art Unit:
	)	
Lakshmi Arunachalam	)	Examiner
	)	
Serial No. 11/980,185	)	
	)	
Filing Date: Oct. 30, 2007	)	
	)	
Title: METHOD AND APPARATUS FOR ENABLING REAL TIME TRANSACTIONS ON A NETWORK	)	

NOTIFICATION OF RELATED LITIGATION

Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

Honorable Commissioner:

This is a notification under M.P.E.P. §2001.06(c) of related litigation of patents in the same family as the present application. A complaints is attached Civil Case CV 08 5149 filed in the United States District Court for the Northern District of California on Nov. 8, 2008. This complaint, filed by Microsoft Corporation, asks for a declaratory Judgment against parent patents 5,778,178, 6,212,556 and 7,340,506.

As of the present time, no material information has arisen from this litigation. If any material information as defined in the above-referenced section arises, the applicant will file a further disclosure.

Respectfully Submitted

Clifford Kraft

Clifford H. Kraft
Reg. No. 35,229
Attorney of Record

CORRESPONDENCE ADDRESS

Clifford H. Kraft
320 Robin Hill Dr.
Naperville, IL 60540

708 528-9092 Tel.
630 393-9114 Fax.

First Class Mailing Certificate: This paper is being filed by United States First Class Mail addressed to Commissioner for Patents, P.O. Box 1450, Alexandria VA. 22313-1450 with sufficient postage on:

Date: Nov. 18, 2008

Signature: Clifford Kraft

Name: Clifford H. Kraft

JOHN D. VANDENBERG
john.vandenberg@klarquist.com
KLARQUIST SPARKMAN, LLP
121 S.W. Salmon Street, Suite 1600
Portland, OR 97204-2988
Telephone: (503) 595-5300
Facsimile: (503) 595-5301

MICHAEL J. BETTINGER (Bar No. 122196)
mike.bettinger@klgates.com
K&L GATES LLP
Four Embarcadero, Suite 1200
San Francisco, CA 94111
Telephone: (415) 882-8200
Facsimile: (415) 882-8220

E-filing

Attorneys for Plaintiff Microsoft Corporation

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

PVT

MICROSOFT CORPORATION, Plaintiff, v. WEBXCHANGE INC., Defendant.	CV Case No. 08 5149 **COMPLAINT FOR DECLARATORY JUDGMENT OF:** **1) PATENT UNENFORCEABILITY,** **2) PATENT INVALIDITY,** **3) NON-INFRINGEMENT**

Plaintiff Microsoft Corporation ("Microsoft") brings this action against WebXchange Inc. ("WebXchange") for a declaratory judgment of patent invalidity, unenforceability and non-infringement.

INTRODUCTION

1. Microsoft publishes and licenses Visual Studio software enabling customers to develop and use a wide variety of computer applications. Visual Studio includes a Web service project template to help developers create "Web services." Web services can be used to allow users to perform interactive, real-time transactions over the World Wide Web and Internet, such as on-line banking and shopping. Microsoft's Visual Studio software provides Web services developers with tools to support use of the Simple Object Access Protocol ("SOAP") in their Web services.

2. WebXchange has placed a cloud over Visual Studio software, Web services, and the SOAP protocol by asserting patents ("the patents in suit") against Microsoft customers for their uses of Web services created using Microsoft's Visual Studio software.

3. WebXchange has alleged a broad scope for these patents, asserting to Microsoft that the patents in suit cover "any real-time transaction on the [Inter]net."

4. WebXchange has already sued three Microsoft customers (Del. D. Ct. Civil Action Nos. 08-131 JJF through 08-133 JJF, hereafter referred to as the "Delaware Lawsuits"). All three have sought indemnification from Microsoft. On information and belief, WebXchange has alleged in the Delaware Lawsuits that use of the SOAP protocol in real-time transactions infringes the patents in suit. On information and belief, WebXchange has also alleged in the Delaware Lawsuits that SOAP-based systems other than those of the defendants in the Delaware Lawsuits infringe the patents.

5. Microsoft is facing potential indemnification demands from additional customers who are sued by WebXchange for patent infringement in the future.

6. Despite WebXchange's broad allegations and despite its suing Microsoft's customers and placing this cloud over Microsoft's Visual Studio software, Web services, and the SOAP

protocol, WebXchange has refused Microsoft's recent entreaties to its counsel to discuss the current disputes and any future potential disputes.

7. WebXchange's strategy of accusing Microsoft's customers one at a time, and refusing to deal with Microsoft, will force Microsoft to expend a disproportionate amount of resources responding to individual customer indemnification demands. WebXchange's strategy, if allowed to continue, will also burden the Courts with a large number of suits when the issues could be resolved in this single suit.

8. The patents in suit are invalid and were obtained by misleading the Patent Office, and no valid claim is infringed by Microsoft's licensing and publication of the Visual Studio software.

9. The relief requested by Microsoft in this action will completely resolve the controversies between WebXchange and Microsoft and between WebXchange and the many Microsoft customers involved in Internet transactions.

THE PARTIES

10. Plaintiff Microsoft is a Washington corporation having its principal place of business at One Microsoft Way, Redmond, Washington 98052. Microsoft has major facilities and thousands of employees in this District.

11. On information and belief, throughout the time period in question in this action, WebXchange's principal place of business has been and still is in this District at 222 Stanford Avenue, Menlo Park, California 94025. The only physical address provided on WebXchange's web page, www.webxchange.com, is an address in this District, 222 Stanford Avenue, Menlo Park, California 94025.

12. Throughout the time period in question in this action, Lakshmi Arunachalam, WebXchange's founder and chief executive officer and the patent applicant on the three patents asserted against Microsoft's customers and challenged in this suit, has been and still is a resident of this District.

JURISDICTION AND VENUE

13. Microsoft realleges and incorporates paragraphs 1 to 12 as if fully set forth herein.

14. This action arises under the patent laws of the United States, Titles 35 of the United States Code. This Court has subject matter jurisdiction pursuant to 28 U.S.C. §§ 1331 and 1338.

15. This Court may enter the declaratory relief sought because this case presents an actual controversy and is within this Court's jurisdiction pursuant to 28 U.S.C. § 2201.

16. WebXchange has claimed to be the owner of U.S. Patent Nos. 5,778,178 (the "'178 Patent"), 6,212,556 (the "'556 Patent"), and 7,340,506 (the "'506 Patent") (collectively, the "Patents in Suit"). True and correct copies of the Patents in Suite are attached hereto as Exhibits A, B, and C.

17. WebXchange has taken the position that the Patents in Suit cover "any real-time transaction on the net," and, on information and belief, has alleged in the Delaware Lawsuits that use of the SOAP protocol in real-time transactions infringes the Patents in Suit.

18. At least thousands of Microsoft customers use Microsoft's Visual Studio software to create software, products and services that offer Web services and enable real-time transactions on the Internet, including transactions that make use of the SOAP protocol. A substantial number of Microsoft customers use software, products and services that were created using Microsoft's Visual Studio software and that offer Web services and enable real-time transactions on the Internet, including transactions that make use of the SOAP protocol. These Microsoft customers are potential targets of WebXchange's expansive infringement allegations. Indeed, on information and belief, WebXchange has specifically alleged in the Delaware Lawsuits that SOAP-based systems other than those of the defendants in the Delaware Lawsuits infringe the Patents in Suit.

19. It is one of Microsoft's goals to protect its customers as much as is reasonably possible against claims of patent infringement citing use of Microsoft's software. As a result, many of Microsoft's software offerings and services are accompanied by Microsoft's agreement to defend and indemnify its customers against various types of patent infringement allegations. Microsoft's willingness to stand behind its offerings is advertised on its web site at, for example, http://www.microsoft.com/iplicensing/IPindemnification.aspx.

20. Customers accused of infringement by WebXchange have sought to have Microsoft defend and indemnify them against WebXchange's allegations.

FIRST CLAIM FOR RELIEF

(Declaratory Judgment of Unenforceability)

21. Microsoft realleges and incorporates paragraphs 1 to 20 as if fully set forth herein.

22. In prosecuting the patent applications which led to the Patents in Suit, patent applicant Arunachalam had a duty of candor and good faith in dealing with the U.S. Patent and Trademark Office ("PTO"), which included a duty to disclose to the PTO all information known to her to be material to patentability.

23. On information and belief, patent applicant Arunachalam drafted and/or reviewed in this judicial district the patent applications which led to the '178, '556, and '506 Patents ("WebXchange Patent Applications"), and made her decisions and communications about the prosecution of these applications—and what information to provide or withhold from the PTO—in this District. Some of the attorneys who assisted Arunachalam with the preparation and prosecution of one or more of the WebXchange Patent Applications were located in Palo Alto, California, in this District.

Arunachalam's Copying from Prior Art
<u>Internet Standards, and Concealment Thereof</u>

24. Concepts and even text in the WebXchange Patent Applications were copied from prior art references.

25. On information and belief, patent applicant Arunachalam either did that copying herself or, at the very least, was aware of such copying prior to or during the prosecution of the WebXchange Patent Applications.

26. Patent applicant Arunachalam did not disclose those prior art references to the PTO or tell the PTO that portions of the WebXchange Patent Applications had been copied from those prior art references.

27. For example, in the early 1990s, the Internet Activities Board published the following as full Internet Standards:

- SMI RFC-1155 ("Structure and Identification of Management Information for TCP/IP-based Internets"), which was published in May 1990,
- MIB II RFC-1213 ("Management Information Base for Network Management of TCP/IP-based Internets"), which was published in March 1991, and
- SNMP RFC-1157 ("A Simple Network Management Protocol (SNMP)"), which was published in May 1990.

28. Many of the concepts presented in the Patents in Suit as being the invention of Arunachalam in fact were published in these Internet Standards in 1990 and 1991.

29. On information and belief, Arunachalam was aware of these Internet Standards by no later than 1994.

30. The WebXchange Patent Applications included concepts and even text copied from these prior art published Internet Standards, but Arunachalam did not disclose that copying to the PTO.

31. For example, in the table below is text from RFC 1213 (published March 1991) and counterpart text from the WebXchange Patent Applications, including issued claims (emphasis added to highlight words and sentences copied verbatim or nearly so from the RFC specification):

RFC 1156 Excerpts	'178 Patent Excerpts
Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using Abstract Syntax Notation One (ASN.1) [8] defined in the [Internet standard] SMI. In particular, each object has a name, a syntax, and an encoding. The name is an object identifier, an administratively assigned name, which specifies an object type. The object type together with an object instance	DOLSIBs are virtual information stores optimized for networking. .... Each object in the DOLSIB has a name, a syntax and an encoding. The name is an administratively assigned object ID specifying an object type. The object type together with the object instance serves to uniquely identify a specific instantiation of the object. ..... The syntax of an object type defines the abstract data structure corresponding to that object type. Encoding

serves to uniquely identify a specific instantiation of the object. For human convenience, we often use a textual string, termed the OBJECT DESCRIPTOR, to also refer to the object type. The syntax of an object type defines the abstract data structure corresponding to that object type. The ASN.1 language is used for this purpose. However, the SMI [12] purposely restricts the ASN.1 constructs which may be used. These restrictions are explicitly made for simplicity. The encoding of an object type is simply how that object type is represented using the object type's syntax. Implicitly tied to the notion of an object type's syntax and encoding is how the object type is represented when being transmitted on the network. The SMI specifies the use of the basic encoding rules of ASN.1 [9], subject to the additional requirements imposed by the SNMP.	of objects defines how the object is represented by the object type syntax while being transmitted over the network. 12. A method for enabling object routing on the World Wide Web, said method for enabling object routing comprising the steps of: creating a virtual information store containing information entries and attributes; .... 15. The method claim 12 wherein said step of associating each of said information entries and said attributes with said object identity further includes the step of storing a name, a syntax and an encoding for each of said object identities. 16. The method in claim 15 wherein said name of said object identity specifies an object type.

32. These Internet Standards, and the fact that content in the WebXchange Patent Applications (including content repeated in some of the claims) was copied from these standards, constituted highly material information that a reasonable PTO Examiner would have considered important in deciding patentability.

33. Yet, Arunachalam intentionally withheld this information from the PTO. On information and belief, Arunachalam withheld this information with the intention of deceiving the PTO into believing that she had invented these concepts, knowing that in fact they had been copied from elsewhere.

Arunachalam's Concealment of Prior Art
Published International Application No. PCT/US96/18165

34. Applicant Arunachalam also intentionally concealed from the PTO other material information.

35. The application that issued as the '556 Patent (the "'556 Continuation-In-Part Application") was filed on April 21, 1999, as a continuation-in-part of an earlier application, but claimed priority to the August 5, 1996 application which issued as the '178 Patent. The application that issued as the '506 Patent (the "'506 Application) was filed on February 23, 2001, as a "divisional" of the application that issued as the '556 Patent.

36. The '556 Continuation-In-Part Application and the '506 Application each contained matter which had not been disclosed in the 1996 application which issued as the '178 Patent ("New Matter").

37. On information and belief, Arunachalam knew in prosecuting the '556 Continuation-In-Part Application and the '506 Application that she was obligated to disclose to the PTO Examiner information published before April 21, 1998 that was material to the patentability of claims not entitled to a filing date earlier than April 21, 1999, and in this District signed a declaration acknowledging that obligation.

38. On October 30, 2000, Arunachalam filed new application claims in the '556 Continuation-In-Part Application, at least some of which were not described in any ancestor application to the '556 Continuation-In-Part Application.

39. One or more claims submitted during the prosecution of the '506 Application and one or more claims of the issued '506 Patent were not described in any application Arunachalam filed prior to the April 21, 1999 parent application.

40. For example, the following '556 Continuation-In-Part Application claims 78 and 84, filed on October 30, 2000, were not entitled to an effective filing date earlier than April 21, 1999:

- "78. (New) The method of claim 67 further including executing the transaction in a distributed computing environment, including creating a plurality of skeleton objects on a computer system remote to the user, registering the plurality of skeleton objects in a name server associated with the remote computer system, and transferring one or more stub objects to a computer system local to the user, wherein the one or more stub objects are derived from the plurality of skeleton objects." and

- "84. (New) The machine-readable medium of claim 80, wherein the instructions further comprise instructions causing the machine to execute the transaction in a distributed computing environment, including instructions to create a plurality of skeleton objects on a computer system remote to the user, register the plurality of skeleton objects in a name server associated with the remote computer system, and transfer one or more stub objects to a computer system local to the user, wherein the one or more stub objects are derived from the plurality of skeleton objects."

41. For example, the following '506 Application claims 74 and 75, filed on February 23, 2001, were not entitled to an effective filing date earlier than April 21, 1999:

- "74. A method comprising: creating a virtual information store containing information entries and attributes; associating each of the information entries and the attributes with a software object identity; describing events and actions of a software object identified by the software object identity using a DOLSIB language construct; and interpreting the DOLSIB language construct describing the events and actions of the software object." and
- "75. The method of claim 74, wherein creating includes creating a virtual information store including the information entries and attributes for each of a plurality of geographically distributed networked software objects including the software object, the virtual information store including a network address for each of the plurality of geographically distributed networked software objects."

42. On information and belief, Arunachalam knew when she filed the four application claims (78, 84, 74 and 75) quoted above in paragraphs 39-40 that their recited methods had not been disclosed in any of her applications filed before April 21, 1999.

43. On May 22, 1997, International Application No. PCT/US96/18165 (naming Lakshmi Arunachalam as the alleged inventor) ("International Application") was published as International Publication No. WO 97/18515. The International Application is entitled, "A method and apparatus for configurable value-added network (VAN) switching and object routing."

44. The disclosure of the International Application is nearly identical to the '178 Patent specification.

45. The International Application was published more than one year before April 21, 1999, and therefore is prior art to any claim of the '556 Patent or '506 Patent entitled to an effective filing date no earlier than April 21, 1999.

46. The International Application was material to the patentability of the four application claims (78, 84, 74 and 75) quoted above in paragraphs 39-40 and the other claims of the '556 Continuation-In-Part Application and '506 Application which are not entitled to an effective filing date before April 21, 1999.

47. On information and belief, Arunachalam knew (prior to issuance of the '556 Patent) of the 1997 publication of her International Application and knew or should have known it was highly material to the patentability of the claims of the '556 Continuation-In-Part Application which incorporated New Matter and the claims of the '506 Application which incorporated New Matter.

48. On information and belief, Arunachalam, with deceptive intent, withheld the International Application from the PTO Examiners in connection with examination of the '556 Continuation-In-Part Application and the '506 Application.

49. The '178, '556 and '506 Patents are unenforceable due to inequitable conduct before the PTO by Arunachalam during prosecution of the '178, '556 and '506 Applications.

50. Microsoft seeks and is entitled to a declaratory judgment that the '178, '556 and '506 Patents are unenforceable.

SECOND CLAIM FOR RELIEF

(Declaratory Judgment of Patent Invalidity – 35 U.S.C. §§ 101 et seq.)

51. Microsoft realleges and incorporates paragraphs 1 to 20 as if fully set forth herein.

52. The Patents in Suit, and each of the claims therein, are invalid for failure to comply the requirements of Title 35 of the United States Code, including without limitation one or more of §§ 101, 102, 103, and 112.

53. Microsoft seeks and is entitled to a declaratory judgment that all claims in the Patents in Suit are invalid.

THIRD CLAIM FOR RELIEF

(Declaratory Judgment of Non-Infringement)

54. Microsoft realleges and incorporates paragraphs 1 to 20 as if fully set forth herein.

55. Microsoft publishes and licenses to customers in this District, and elsewhere, Visual Studio software.

56. WebXchange has asserted that use of Web services created using Microsoft's Visual Studio software, including use of Web services that make use of the SOAP protocol for real-time transactions, infringes the Patents in Suit.

57. Microsoft's publication and licensing of its Visual Studio software does not infringe any valid claim of the Patents in Suit.

58. Microsoft's customers' use of Web services created using Microsoft's Visual Studio software, including use of Web services that make use of the SOAP protocol for real-time transactions, does not infringe any valid claim of the Patents in Suit.

59. Microsoft seeks and is entitled to a declaratory judgment that its publication and licensing of its Visual Studio software does not infringe any valid claim of the Patents in Suit and that Microsoft's customers' use of Web services created using Microsoft's Visual Studio software, including use of Web services that make use of the SOAP protocol for real-time transactions, does not infringe any valid claim of the Patents in Suit.

PRAYER FOR RELEF

WHEREFORE, Microsoft requests entry of judgment in its favor and against WebXchange as follows:

A. For a declaration of this Court that the Patents in Suit, and each of the claims therein, are invalid;

B. For a declaration of this Court that the Patents in Suit are unenforceable;

C. For a declaration of this Court that Microsoft's publication and licensing of its Visual Studio software does not infringe any valid claim of the Patents in Suit;

D. For a declaration of this Court that Microsoft's customers' use of Web services created using Microsoft's Visual Studio software, including use of Web services that make use of the SOAP protocol for real-time transactions, does not infringe any valid claim of the Patents in Suit;

E. For costs and reasonable attorneys' fees incurred in connection with this action; and

F. For such other and further relief as the court deems just.

Dated this 11th day of November, 2008.

By: ______________________________

Michael J. Bettinger (Bar No. 122196)
mike.bettinger@klgates.com
K&L Gates LLP
Four Embarcadero, Suite 1200
San Francisco, CA 94111
Telephone: (415) 882-8200
Facsimile: (415) 882-8220

John D. Vandenberg
john.vandenberg@klarquist.com
KLARQUIST SPARKMAN, LLP
121 S.W. Salmon Street, Suite 1600
Portland, OR 97204-2988
Telephone: (503) 595-5300
Facsimile: (503) 595-5301

Attorneys for Plaintiff Microsoft Corporation

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	)	Art Unit: 2155
Lakshmi Arunachalam	)	Examiner:
Serial No. 11/980,185	)	
Filing Date: Oct 30, 2007	)	
Title: METHOD AND APPARAUTS FOR ENABLING REAL TIME TRANSACTIONS ON A NETWORK	)	

INFORMATION DISCLOSURE STATEMENT

Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

Honorable Commissioner:

In accordance with 37 C.F.R. §1.97, please accept this Information Disclosure Statement and copies of any non-US patent art.

COMMENTS

It is believed that this disclosure complies with 37 C.F.R. §1.56 and 1.98 and M.P.E.P. §2000. This disclosure statement should not be construed as a representation that a search has been made or that no other material information as defined in 37 C.F.R. §1.56(a) exists. A copy of each non-US patent reference is being supplied. Some references may contain marks; no significance should be attached to these.

Respectfully submitted

Clifford Kraft

Cifford H. Kraft
Reg. No. 35,229
Attorney of Record

CORRESPONDENCE ADDRESS **CUSTOMER NUMBER: 000074642**

Clifford H. Kraft
320 Robin Hill Dr.
Naperville, IL 60540

(708) 528-9092

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to Commissioner for Patents, P.O. Box 1450 Alexandria, VA 22313-1450 with sufficient postage.

On: OCT. 23, 2008

By: Clifford Kraft

Name: Clifford H. Kraft

PTO/SB/08a (09-08)
Approved for use through 10/31/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	11/980,185
Filing Date	10-30-2007
First Named Inventor	Lakshmi Arunachalam
Art Unit	
Examiner Name	
Attorney Docket Number	

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US- 5,475,819	12-12-1995	Miller et al.	709/203
		US- 5,859,978	01-12-1999	Sonderegger et al.	709/226
		US- 6,249,291	06-19-2001	Popp et al.	345/473
		US- 5,347,632	09-13-2004	Filepp et al.	709/202
		US- 6,092,053	07-18-2000	Boesch et al.	705/26
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]-Number[4]-Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional). [2] See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(c) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/980,185	10/30/2007	Lakshmi Arunachalam	

CONFIRMATION NO. 5863

Clifford Kraft
320 Robin Hill Dr.
Naperville, IL60540

Title: Method and apparatus for enabling real-time bi-directional transactions on a network

Publication No. US-2008-0091801-A1
Publication Date: 04/17/2008

NOTICE OF PUBLICATION OF APPLICATION

The above-identified application will be electronically published as a patent application publication pursuant to 37 CFR 1.211, et seq. The patent application publication number and publication date are set forth above.

The publication may be accessed through the USPTO's publically available Searchable Databases via the Internet at www.uspto.gov. The direct link to access the publication is currently http://www.uspto.gov/patft/.

The publication process established by the Office does not provide for mailing a copy of the publication to applicant. A copy of the publication may be obtained from the Office upon payment of the appropriate fee set forth in 37 CFR 1.19(a)(1). Orders for copies of patent application publications are handled by the USPTO's Office of Public Records. The Office of Public Records can be reached by telephone at (703) 308-9726 or (800) 972-6382, by facsimile at (703) 305-8759, by mail addressed to the United States Patent and Trademark Office, Office of Public Records, Alexandria, VA 22313-1450 or via the Internet.

In addition, information on the status of the application, including the mailing date of Office actions and the dates of receipt of correspondence filed in the Office, may also be accessed via the Internet through the Patent Electronic Business Center at www.uspto.gov using the public side of the Patent Application Information and Retrieval (PAIR) system. The direct link to access this status information is currently http://pair.uspto.gov/. Prior to publication, such status information is confidential and may only be obtained by applicant using the private side of PAIR.

Further assistance in electronically accessing the publication, or about PAIR, is available by calling the Patent Electronic Business Center at 1-866-217-9197.

Pre-Grant Publication Division, 703-605-4283

IFW

IN THE PATENT AND TRADEMARK OFFICE

In re application of:	Art Unit: 2154
Lakshmi Arunachalam	Examiner:
Serial No.: 11/980,185	
Filing Date: Oct. 30, 2007	
Title: METHOD AND APPARATUS FOR ENABLING REAL-TIME BI-DIRECTIONAL TRANSACTIONS ON A NEWWORK	

INFORMATION DISCLOSURE STATEMENT

Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

Honorable Commissioner:

In accordance with 37 C.F.R. §1.97, please accept this Information Disclosure Statement, cross-reference to co-pending applications and copies of any non-US patent art.

COMMENTS

It is believed that this disclosure complies with 37 C.F.R. §1.56 and 1.98 and M.P.E.P. §2000. This disclosure statement should not be construed as a representation that a search has been made or that no other material information as defined in 37 C.F.R. §1.56(a) exists. A copy of each non-US patent reference is being supplied. Some references may contain marks; no significance should be attached to these.

Respectfully submitted

Clifford Kraft

Cifford H. Kraft
Reg. No. 35,229
Attorney of Record

CORRESPONDENCE ADDRESS

Clifford H. Kraft
320 Robin Hill Dr.
Naperville, IL 60540

(708) 528-9092

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to Commissioner for Patents, P.O. Box 1450 Alexandria, VA 22313-1450 with sufficient postage.

On: APRIL 13, 2008

By: Clifford Kraft

Name: Clifford H. Kraft

APR 16 2008 PATENT & TRADEMARK OFFICE

PTO/SB/08A (01-08)
Approved for use through 04/30/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 8

Complete if Known	
Application Number	11/980,185
Filing Date	Oct. 30, 2007
First Named Inventor	Lakshmi Arunachalam
Art Unit	
Examiner Name	
Attorney Docket Number	

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Name	Publication Date MM-DD-YYYY	Author if known
	1	Dr. GUI on Components, COM, and ATL Part 1: You're Gonna Do COM?...	2-2-1998	http://msdn.microsoft.com/library/welcome/dsmsdn/msdn-drguion020298.htm
		Part 2: Basics of COM	2-9-1998	" " "
		Part 3: Getting Objects and Interfaces	2-23-1998	" " "
		Part 4: The Class Object and Class Factory	3-2-1998	" " "
		Part 5: Implementing an Object	3-30-1998	" " "
		Part 6: Using our COM Object in Visual Basic...	4-27-1998	" " "
		Part 7: Using our Object from Visual C++	5-29-1998	" " "
		Part 8: Get Smart! Using our COM Object...	7-30-1998	" " "
	2	Microsoft COM.NEWS Dr. GUI's Gentle Guide to COM	~11-1-1999	http://www.microsoft.com/Com/news/drgui.asp

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]-Number[4]-Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional). [2] See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

PTO/SB/08A (01-08)
Approved for use through 04/30/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 2 of 8

Complete if Known	
Application Number	11/980,185
Filing Date	Oct. 30, 2007
First Named Inventor	Lakshmi Arunachalam
Art Unit	
Examiner Name	
Attorney Docket Number	

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Name	Publication Date MM-DD-YYYY	Author if known
	3	Taking the Splash Diving into ISAPI Programming	1/1997	Christian Gross http://www.microsoft.com/mind/0197/isapi-htm
	4	Chapter 1, NSAPI Basics	12/22/1997	http://developer.netscape.com/docs/manuals/enterprise/nsapi/svrop.htm
	5	The Common Gateway Interface	Retrieved 5/22/2001	http://hoohoo.ncsa.uiuc.edu/cgi/primer.html
	6	Open Market Content-Driven eBusiness Solutions	Retrieved 5/15/2001	http://www.openmarket.com/cgi-bin/gx.cgi/AppLogic+FTContentServer?pagename=FutureTense/Apps/Xcelerate/View&c=Collec...
	7	Open Market Content Server	5/15/2001	http://www.openmarket.com/cgi-bin/gx.cgi/AppLogic+FTContentServer?pagename=FutureTense/Apps/Xcelerate/Render&c=Artic...

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]-Number[4]-Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1]Applicant's unique citation designation number (optional). [2]See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3]Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4]For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5]Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6]Applicant is to place a check mark here if English language Translation is attached.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 3 of 8

Complete if Known	
Application Number	11/980,185
Filing Date	Oct. 30, 2007
First Named Inventor	Lakshmi Arunachalam
Art Unit	
Examiner Name	
Attorney Docket Number	

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Name	Publication Date MM-DD-YYYY	Author if known	
	8	Open Market Content Centre	Retrieved 5/15/2001	http://www.openmarket.com/cgi-bin/gx.cgi/AppLogic+FTContentServer?pagename=FutureTense/Apps/Xcelerate/Render&c=Artic...	
	9	OpenMarket Integration Centre	5/15/2001	"	"
	10	OpenMarket Personalization Centre	5/15/2001	"	"
	11	OpenMarket Catalog Centre	5/15/2001	"	"
	12	OpenMarket Marketing Studio	5/15/2001	"	"
	13	OpenMarket Satellite Server	5/15/2001	"	"

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]-Number[4]-Kind Code[5] (*if known*)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1]Applicant's unique citation designation number (optional). [2]See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3]Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4]For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5]Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6]Applicant is to place a check mark here if English language Translation is attached.

PTO/SB/08A (01-08)
Approved for use through 04/30/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 4 of 8

Complete if Known	
Application Number	11/980,185
Filing Date	Oct. 30, 2007
First Named Inventor	Lakshmi Arunachalam
Art Unit	
Examiner Name	
Attorney Docket Number	

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Name	Publication Date MM-DD-YYYY	Author if known	
	14	OpenMarket Commerce Products	5/15/2001 Retrieved?	http://www.openmarket.com/cgi-bin/gx.cgi/Apologic+FTContentServer?pagename=FutureTense/Apps/Xcelerate/...	
	15	OpenMarket Transact	5/15/2001	"	" "
	16	OpenMarket Shopsite	5/15/2001	"	" "
	17	OpenMarket Open Exchange Shop Site 5.0	5/15/2001	"	" "
	18	OpenMarket Wireless Solutions, an OpenMarket eBusiness Solution Brief	2/13/2001	http://www.openmarket.com/	

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] -Number[4] -Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional). [2] See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

PTO/SB/08A (01-08)
Approved for use through 04/30/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 5 of 8

Complete if Known	
Application Number	11/980,185
Filing Date	Oct. 30, 2007
First Named Inventor	Lakshmi Arunachalam
Art Unit	
Examiner Name	
Attorney Docket Number	

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Name	Publication Date MM-DD-YYYY	Author if known	
	19	Open Market Portal Solutions an OpenMarket eBusiness Solution Brief	2/21/2001	http://www.openmarket.com/	
	20	CyberCash Inc. - The E-Commerce Leader in Payment Solutions - B2B	1996 Retrieved 5/23/2001	http://www.cybercash.com/	
	21	CyberCash Products	5/23/2001	http://www.cybercash.com/products/	
	23	CyberCash Cash Register - Online Secure Payment Service	5/23/2001	http://www.cybercash.com/cashregister/	
	22	Cybercash ICVERIFY 2.5 Upgrade	5/23/2001	http://www.cybercash.com/icverify/upgrade.html	

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1]Applicant's unique citation designation number (optional). [2]See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3]Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4]For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5]Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6]Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

PTO/SB/08A (01-08)
Approved for use through 04/30/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 6 of 8

Complete if Known	
Application Number	11/980,185
Filing Date	Oct. 30, 2007
First Named Inventor	Lakshmi Arunachalam
Art Unit	
Examiner Name	
Attorney Docket Number	

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Name	Publication Date MM-DD-YYYY	Author if known	
	24	Cybercash Cash Register - How It Works	Retrieved 5/23/2001	http://www.cybercash.com/cashregister/howitworks.html	
	25	Cybercash Cash Register - Industry Leading Features	5/23/2001	http://www.cybercash.com/cashregister/features.html	
	26	Cybercash Cash Register - Why Choose Cash Register?	5/23/2001	http://www.cybercash.com/cashregister/why.html	
	27	Cybercash Cash Register - Online Secure Payment Service	2000	http://webdata.cybercash.com/demos/	
	28	Cybercash Web Authorize - Enterprise and Hosting Payment Processing	Retrieved 5/23/2001	http://www.cybercash.com/webauthorize/	

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]-Number[4]-Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional). [2] See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 7 of 8

Complete if Known	
Application Number	11/980,185
Filing Date	Oct. 30, 2007
First Named Inventor	Lakshmi Arunachalam
Art Unit	
Examiner Name	
Attorney Docket Number	

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Name	Publication Date MM-DD-YYYY	Author if known
	29	Cybercash B2B Payment Services	Retrieved: 5/23/2001	http://www.cybercash.com/b2b/
	30	Cybercash Fraud Patrol Service	5/23/2001	http://www.cybercash.com/fraudpatrol/
	34	Cybercash PCAuthorize - Payment Software for Brick-and-Mortar Merchants	5/23/2001	http://www.cybercash.com/pcauthorize/
	35	Microsoft Component Services - Server Operating System - A Technology Overview	Dated 8/15/98 Retrieved 5/22/2001	http://www.microsoft.com/com/wpaper/compsvcs.asp
	36	iPIN Home	Retrieved 5/23/2001	http://www.ipin.com/
	37	iPIN Company Info	" "	http://www.ipin.com/01comp.html

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]-Number[4]-Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1]Applicant's unique citation designation number (optional). [2]See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3]Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4]For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5]Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6]Applicant is to place a check mark here if English language Translation is attached.

PTO/SB/08A (01-08)
Approved for use through 04/30/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 3 of 8

Complete if Known	
Application Number	11/980,185
Filing Date	Oct. 30, 2007
First Named Inventor	Lakshmi Arunachalam
Art Unit	
Examiner Name	
Attorney Docket Number	

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Name	Publication Date MM-DD-YYYY	Author if known	
	38	IPIN Products - The IPIN Approach	Retrieved 5/23/2001	http://www.ipin.com/02prod.html	
	39	IPIN Products - Technology	" "	http://www.ipin.com/02prod-tech.html	
	40	IPIN Products - Solutions	" "	http://www.ipin.com/02prod-solution.html	
	41	IPIN Products - Service Options	" "	http://www.ipin.com/02prod-service.html	
	42	IPIN Partners	" "	http://www.ipin.com/03part.html	
	31	Cybercash Fraud Patrol How It Works	" "	http://www.cybercash.com/fraudpatrol/howitworks.html	

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]-Number[4]-Kind Code[5] *(if known)*	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1]Applicant's unique citation designation number (optional). [2]See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5]Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6]Applicant is to place a check mark here if English language Translation is attached.

IPW

IN THE PATENT AND TRADEMARK OFFICE

In re application of:	)	Art Unit: 2154
Lakshmi Arunachalam	)	Examiner:
Serial No.: 11/980,185	)	
Filing Date: Oct. 30, 2007	)	
Title: METHOD AND APPARATUS FOR ENABLING REAL-TIME BI-DIRECTIONAL TRANS-ACTIONS ON A NEWWORK	)	

INFORMATION DISCLOSURE STATEMENT

Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

Honorable Commissioner:

In accordance with 37 C.F.R. §1.97, please accept this Information Disclosure Statement, cross-reference to co-pending applications and copies of any non-US patent art.

COMMENTS

It is believed that this disclosure complies with 37 C.F.R. §1.56 and 1.98 and M.P.E.P. §2000. This disclosure statement should not be construed as a representation that a search has been made or that no other material information as defined in 37 C.F.R. §1.56(a) exists. A copy of each non-US patent reference is being supplied. Some references may contain marks; no significance should be attached to these.

Respectfully submitted

Clifford Kraft

Cifford H. Kraft
Reg. No. 35,229
Attorney of Record

CORRESPONDENCE ADDRESS

Clifford H. Kraft
320 Robin Hill Dr.
Naperville, IL 60540

(708) 528-9092

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to Commissioner for Patents, P.O. Box 1450 Alexandria, VA 22313-1450 with sufficient postage.

On: Feb. 5, 2008

By: Clifford Kraft

Name: Clifford H. Kraft

OIPE IAP88 FEB 07 2008 PATENT & TRADEMARK OFFICE

PTO/SB/08A (01-08)
Approved for use through 01/31/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet | of

Complete if Known	
Application Number	11/980,185
Filing Date	10-30-2007
First Named Inventor	Lakshmi Arunachalm
Art Unit	2154
Examiner Name	
Attorney Docket Number	

	1	Lichty, Tom, "America Online tour Guide", MacIntosh Edition, Version 2, Preface, Chapter 1, Ventanna Press, 1992	
	2	Lichty, Tom, "America Online tour Guide", MacIntosh Edition, Version 2, Preface, Chapter 3, Ventanna Press, 1992	
	3	Lichty, Tom, "America Online tour Guide", MacIntosh Edition, Version 2, Preface, Chapter 8, Ventanna Press, 1992	
	4	Lichty, Tom, "America Online tour Guide", MacIntosh Edition, Version 2, Preface, Chapter 10, Ventanna Press, 1992	

IFW

IN THE PATENT AND TRADEMARK OFFICE

In re application of:	)	Art Unit: 2154
Lakshmi Arunachalam	)	Examiner:
Serial No.: 11/980,185	)	
Filing Date: Oct. 30, 2007	)	
Title: METHOD AND APPARATUS FOR ENABLING REAL-TIME BI-DIRECTIONAL TRANS-ACTIONS ON A NEWWORK	)	

INFORMATION DISCLOSURE STATEMENT

Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

Honorable Commissioner:

In accordance with 37 C.F.R. §1.97, please accept this Information Disclosure Statement, cross-reference to co-pending applications and copies of any non-US patent art.

CROSS-REFERENCE TO CO-PENDING APPLICATIONS

The following US applications are co-pending naming the same inventor:

09/863,704 filed 5-23-2001
09/792,323 filed 2-23-2001

COMMENTS

It is believed that this disclosure complies with 37 C.F.R. §1.56 and 1.98 and M.P.E.P. §2000. This disclosure statement should not be construed as a representation that a search has been made or that no other material information as defined in 37 C.F.R. §1.56(a) exists. A copy of each non-US patent reference is being supplied. Some references may contain marks; no significance should be attached to these.

Respectfully submitted

Clifford Kraft

Cifford H. Kraft
Reg. No. 35,229
Attorney of Record

CORRESPONDENCE ADDRESS

Clifford H. Kraft
320 Robin Hill Dr.
Naperville, IL 60540

(708) 528-9092

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to Commissioner for Patents, P.O. Box 1450 Alexandria, VA 22313-1450 with sufficient postage.

On: FEB. 1, 2008

By: Clifford Kraft

Name: Clifford H. Kraft

PTO/SB/08A (01-08)
Approved for use through 01/31/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 4

Complete if Known	
Application Number	11/980,185
Filing Date	10-30-2007
First Named Inventor	Lakshmi Arunachalm
Art Unit	
Examiner Name	
Attorney Docket Number	

DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	
		US-5,491,800	02-1996	GOLDSMITH et al.	
		US-5,577,251	11-1996	HAMILTON et al	
		US-6,101,527	08-2000	LEJEUNE et al	
		US-6,249,291	06-2001	POPP et al	
		US-6,553,427	04-2003	CHANG et al	
		US-5,710,887	01-1998	CHELLIAH et al	
		US-5,455,903	10-3-95	JOLISSAINT et al	
		US-5,715,314	2-3-98	PAYNE et al	
		US-5,771,354	6-23-98	CRAWFORD	
		US-5,901,228	5-4-99	CRAWFORD	
		US-6,014,651	1-11-2000	CRAWFORD	
		US-6,327,579	12-4-01	CRAWFORD	
		US-6,411,943	6-25-02	CRAWFORD	
		US-7,080,051	7-18-06	CRAWFORD	
		US-5,870,724	2-99	LAWLOR et al	
		US-5,677,708	10-97	MATHEWS et al	
		US-~~5,455,903~~	~~10-3-95~~	~~JOLISS.~~	
		US-5,442,771	8-15-95	FILEPP et al	
		US-5,347,632	9-13-94	FILEPP et al	

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]-Number[4]-Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1]Applicant's unique citation designation number (optional). [2]See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3]Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4]For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5]Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6]Applicant is to place a check mark here if English language Translation is attached.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 2 of 4

Complete if Known	
Application Number	11/980,185
Filing Date	10-30-2007
First Named Inventor	Lakshmi Arunachalm
Art Unit	
Examiner Name	
Attorney Docket Number	

DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	
		US- 5,239,662	8-93	DANIELSON et al	
		US- 6,055,514	4-00	WREN	
		US- 5,892,821	4-99	TURNER	
		US- 5,828,666	10-98	FOCSANEANU et al	
		US- 5,557,780	9-96	EDWARDS et al	
		US- 5,537,464	7-96	LEWIS et al	
		US- 5,148,474	9-92	HARALAMBOPOULOS et al.	
		US- 5,812,779	9-98	CISCON et al	
		US- 5,794,234	8-98	CHURCH et al	
		US- 5,878,403	3-99	DEFRANCESCO et al	
		US- 5,893,076	4-99	HAFNER et al	
		US- 5,870,473	2-99	BOESCH et al	
		US- 6,212,634	4-01	GEER JR et al	
		US- 6,049,785	4-00	GIFFORD	
		US- 5,909,492	6-99	PAYNE et al	
		US- 5,724,424	3-98	GIFFORD	
		US- 5,715,314	2-98	PAYNE et al	
		US- 6,128,315	10-00	TAKEUCHI	
		US- 6,185,609	2-01	RANGARAJAN et al	

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]-Number[4]-Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional). [2] See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

PTO/SB/08A (01-08)
Approved for use through 01/31/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 3 of 4

Complete if Known	
Application Number	11/980,185
Filing Date	10-30-2007
First Named Inventor	Lakshmi Arunachalm
Art Unit	
Examiner Name	
Attorney Docket Number	

DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	
		US-5,864,866	1-99	HENCKEL et al	
		US-5,913,061	6-99	GUPTA et al	
		US-6,092,053	7-00	BOESCH et al	
		US-6,192,250	2-01	BUSKENS et al	
		US-6,049,819	4-00	BUCKLE et al	
		US-5,329,619	7-94	PAGE et al	
		US-5,965,509	9-99	KEVNER	
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]-Number[4]-Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1]Applicant's unique citation designation number (optional). [2]See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3]Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4]For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5]Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6]Applicant is to place a check mark here if English language Translation is attached.

PTO/SB/08A (01-08)
Approved for use through 01/31/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 4 of 4

Complete if Known	
Application Number	11/980,185
Filing Date	10-30-2007
First Named Inventor	Lakshmi Arunachalm
Art Unit	
Examiner Name	
Attorney Docket Number	

	1	LAMOND, Keith, "Credit Card Transactions Real World and Online" http://www.virtualshcool.edu/mon/ElectronProperty/klamond/credit_card/htm, pp 1-16, 1996	
	2	COX, Benjamin et al., "Netbill Security and Transaction Protocol", Carnegie Millon University, Pittsburgh, PA 15212-3890	
	3	"Tymnet", Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/tymnet, May 2007.	
	4	HICKEY, "Shopping at Home: One Modem Line, No Waiting", Home PC, Dec. 1, 1994, p. 307, Dialog, File 647, Acc# 01038162	
	5	LANG, "Cashing in: The Rush is on to Buy and Sell on the Internet but Conflicting Schemes Leave Marketers on Sidelines for Now", Advertising Age, Dec. 19, 1994, p. 11, Dialog File 16, Acc# 05419137	
	6	BANKS, Michael A., "America Online: A Graphics-based Success", Link-Up, Jan/Feb 1992	
	7	"Hot Jave", Wikipeda, the free encycliopedia, http://en.wikipedia.org/wiki/HotJava, May 2007	

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING or 371(c) DATE	GRP ART UNIT	FIL FEE REC'D	ATTY.DOCKET.NO	TOT CLAIMS	IND CLAIMS
11/980,185	10/30/2007	2154	3880		110	13

CONFIRMATION NO. 5863

Clifford Kraft
320 Robin Hill Dr.
Naperville, IL 60540

UPDATED FILING RECEIPT

OC000000027647799

Date Mailed: 01/11/2008

Receipt is acknowledged of this non-provisional patent application. The application will be taken up for examination in due course. Applicant will be notified as to the results of the examination. Any correspondence concerning the application must include the following identification information: the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. **If an error is noted on this Filing Receipt, please write to the Office of Initial Patent Examination's Filing Receipt Corrections. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections**

Applicant(s)

Lakshmi Arunachalam, Menlo Park, CA;

Power of Attorney:
Clifford Kraft--35229

Domestic Priority data as claimed by applicant

This application is a CIP of 09/792,323 02/23/2001
which is a CIP of 08/879,958 06/20/1997 PAT 5,987,500
which is a DIV of 08/700,726 08/05/1996 PAT 5,778,178
and claims benefit of 60/006,634 11/13/1995

Foreign Applications

If Required, Foreign Filing License Granted: 11/28/2007

The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is **US 11/980,185**

Projected Publication Date: 04/17/2008

Non-Publication Request: No

Early Publication Request: No
**** SMALL ENTITY ****

Title

Method and apparatus for enabling real-time bi-directional transactions on a network

Preliminary Class

709

PROTECTING YOUR INVENTION OUTSIDE THE UNITED STATES

Since the rights granted by a U.S. patent extend only throughout the territory of the United States and have no effect in a foreign country, an inventor who wishes patent protection in another country must apply for a patent in a specific country or in regional patent offices. Applicants may wish to consider the filing of an international application under the Patent Cooperation Treaty (PCT). An international (PCT) application generally has the same effect as a regular national patent application in each PCT-member country. The PCT process **simplifies** the filing of patent applications on the same invention in member countries, but **does not result** in a grant of "an international patent" and does not eliminate the need of applicants to file additional documents and fees in countries where patent protection is desired.

Almost every country has its own patent law, and a person desiring a patent in a particular country must make an application for patent in that country in accordance with its particular laws. Since the laws of many countries differ in various respects from the patent law of the United States, applicants are advised to seek guidance from specific foreign countries to ensure that patent rights are not lost prematurely.

Applicants also are advised that in the case of inventions made in the United States, the Director of the USPTO must issue a license before applicants can apply for a patent in a foreign country. The filing of a U.S. patent application serves as a request for a foreign filing license. The application's filing receipt contains further information and guidance as to the status of applicant's license for foreign filing.

Applicants may wish to consult the USPTO booklet, "General Information Concerning Patents" (specifically, the section entitled "Treaties and Foreign Patents") for more information on timeframes and deadlines for filing foreign patent applications. The guide is available either by contacting the USPTO Contact Center at 800-786-9199, or it can be viewed on the USPTO website at http://www.uspto.gov/web/offices/pac/doc/general/index.html.

For information on preventing theft of your intellectual property (patents, trademarks and copyrights), you may wish to consult the U.S. Government website, http://www.stopfakes.gov. Part of a Department of Commerce initiative, this website includes self-help "toolkits" giving innovators guidance on how to protect intellectual property in specific countries such as China, Korea and Mexico. For questions regarding patent enforcement issues, applicants may call the U.S. Government hotline at 1-866-999-HALT (1-866-999-4158).

LICENSE FOR FOREIGN FILING UNDER

Title 35, United States Code, Section 184

Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Bureau of Industry and Security, Department of Commerce (15 CFR parts 730-774); the Office of Foreign AssetsControl, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/980,185	10/30/2007	Lakshmi Arunachalam	

CONFIRMATION NO. 5863

FORMALITIES LETTER

OC000000027015761

Clifford Kraft
320 Robin Hill Dr.
Naperville, IL 60540

Date Mailed: 12/05/2007

NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION

FILED UNDER 37 CFR 1.53(b)

Filing Date Granted

Items Required To Avoid Abandonment:

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given **TWO MONTHS** from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The statutory basic filing fee is missing.
  Applicant must submit ***$155*** *to complete the basic filing fee for a small entity.*

The applicant needs to satisfy supplemental fees problems indicated below.

The required item(s) identified below must be timely submitted to avoid abandonment:

- Additional claim fees of **$3300** as a small entity, including any required multiple dependent claim fee, are required. Applicant must submit the additional claim fees or cancel the additional claims for which fees are due.
- To avoid abandonment, a surcharge (for late submission of filing fee, search fee, examination fee or oath or declaration) as set forth in 37 CFR 1.16(f) of **$65** for a small entity in compliance with 37 CFR 1.27, must be submitted with the missing items identified in this notice.

SUMMARY OF FEES DUE:

Total additional fee(s) required for this application is **$3880** for a small entity
- **$155** Statutory basic filing fee.
- **$65** Surcharge.
- The application search fee has not been paid. Applicant must submit **$255** to complete the search fee.
- The application examination fee has not been paid. Applicant must submit **$105** to complete the examination fee for a small entity in compliance with 37 CFR 1.27.
- Total additional claim fee(s) for this application is **$3300**
  - **$1050** for **10** independent claims over 3.
  - **$2250** for **90** total claims over 20.

01/03/2008 AAHMADI 00000032 11980185

05 FC:2201 1050.00 OP
06 FC:2202 2250.00 OP

01/03/2008 AAHMADI 00000032 11980185

01 FC:2011 155.00 OP
02 FC:2111 255.00 OP
03 FC:2311 105.00 OP
04 FC:2051 65.00 OP

Replies should be mailed to:

Mail Stop Missing Parts
Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

Registered users of EFS-Web may alternatively submit their reply to this notice via EFS-Web.
https://sportal.uspto.gov/authenticate/AuthenticateUserLocalEPF.html

For more information about EFS-Web please call the USPTO Electronic Business Center at **1-866-217-9197** or visit our website at http://www.uspto.gov/ebc.

If you are not using EFS-Web to submit your reply, you must include a copy of this notice.

/tvo/

Office of Initial Patent Examination (571) 272-4000 or 1-800-PTO-9199

This paper is being submitted by United States First Class Mail with sufficient postage addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria VA 22313-1450 on:

Date: DEC. 30, 2007

Signature: Clifford Kraft

Name: **Clifford H. Kraft**

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/980,185	10/30/2007	Lakshmi Arunachalam	

CONFIRMATION NO. 5863

Clifford Kraft
320 Robin Hill Dr.
Naperville, IL 60540

FORMALITIES LETTER

OC000000027015761

Date Mailed: 12/05/2007

NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION

FILED UNDER 37 CFR 1.53(b)

Filing Date Granted

Items Required To Avoid Abandonment:

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given **TWO MONTHS** from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The statutory basic filing fee is missing.
  *Applicant must submit $***155** *to complete the basic filing fee for a small entity.*

The applicant needs to satisfy supplemental fees problems indicated below.

The required item(s) identified below must be timely submitted to avoid abandonment:

- Additional claim fees of $**3300** as a small entity, including any required multiple dependent claim fee, are required. Applicant must submit the additional claim fees or cancel the additional claims for which fees are due.
- To avoid abandonment, a surcharge (for late submission of filing fee, search fee, examination fee or oath or declaration) as set forth in 37 CFR 1.16(f) of $**65** for a small entity in compliance with 37 CFR 1.27, must be submitted with the missing items identified in this notice.

SUMMARY OF FEES DUE:

Total additional fee(s) required for this application is $**3880** for a small entity

- $**155** Statutory basic filing fee.
- $**65** Surcharge.
- The application search fee has not been paid. Applicant must submit $**255** to complete the search fee.
- The application examination fee has not been paid. Applicant must submit $**105** to complete the examination fee for a small entity in compliance with 37 CFR 1.27.
- Total additional claim fee(s) for this application is $**3300**
  - $**1050** for **10** independent claims over 3.
  - $**2250** for **90** total claims over 20.

Replies should be mailed to:

Mail Stop Missing Parts
Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

Registered users of EFS-Web may alternatively submit their reply to this notice via EFS-Web.
https://sportal.uspto.gov/authenticate/AuthenticateUserLocalEPF.html

For more information about EFS-Web please call the USPTO Electronic Business Center at **1-866-217-9197** or visit our website at http://www.uspto.gov/ebc.

If you are not using EFS-Web to submit your reply, you must include a copy of this notice.

/tvo/

Office of Initial Patent Examination (571) 272-4000 or 1-800-PTO-9199

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING or 371(c) DATE	GRP ART UNIT	FIL FEE REC'D	ATTY.DOCKET.NO	TOT CLAIMS	IND CLAIMS
11/980,185	10/30/2007	2154	0.00		110	13

CONFIRMATION NO. 5863

FILING RECEIPT

OC000000027015760

Clifford Kraft
320 Robin Hill Dr.
Naperville, IL 60540

Date Mailed: 12/05/2007

Receipt is acknowledged of this non-provisional patent application. The application will be taken up for examination in due course. Applicant will be notified as to the results of the examination. Any correspondence concerning the application must include the following identification information: the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. **If an error is noted on this Filing Receipt, please write to the Office of Initial Patent Examination's Filing Receipt Corrections. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections**

Applicant(s)

Lakshmi Arunachalam, Menlo Park, CA;

Power of Attorney:
Clifford Kraft--35229

Domestic Priority data as claimed by applicant

This application is a CIP of 09/792,323 02/23/2001
which is a CIP of 08/879,958 06/20/1997 PAT 5,987,500
which is a DIV of 08/700,726 08/05/1996 PAT 5,778,178
and claims benefit of 60/006,634 11/13/1995

Foreign Applications

If Required, Foreign Filing License Granted: 11/28/2007

The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is **US 11/980,185**

Projected Publication Date: To Be Determined - pending completion of Missing Parts

Non-Publication Request: No

Early Publication Request: No
**** SMALL ENTITY ****

Title

Method and apparatus for enabling real-time bi-directional transactions on a network

Preliminary Class

709

PROTECTING YOUR INVENTION OUTSIDE THE UNITED STATES

Since the rights granted by a U.S. patent extend only throughout the territory of the United States and have no effect in a foreign country, an inventor who wishes patent protection in another country must apply for a patent in a specific country or in regional patent offices. Applicants may wish to consider the filing of an international application under the Patent Cooperation Treaty (PCT). An international (PCT) application generally has the same effect as a regular national patent application in each PCT-member country. The PCT process **simplifies** the filing of patent applications on the same invention in member countries, but **does not result** in a grant of "an international patent" and does not eliminate the need of applicants to file additional documents and fees in countries where patent protection is desired.

Almost every country has its own patent law, and a person desiring a patent in a particular country must make an application for patent in that country in accordance with its particular laws. Since the laws of many countries differ in various respects from the patent law of the United States, applicants are advised to seek guidance from specific foreign countries to ensure that patent rights are not lost prematurely.

Applicants also are advised that in the case of inventions made in the United States, the Director of the USPTO must issue a license before applicants can apply for a patent in a foreign country. The filing of a U.S. patent application serves as a request for a foreign filing license. The application's filing receipt contains further information and guidance as to the status of applicant's license for foreign filing.

Applicants may wish to consult the USPTO booklet, "General Information Concerning Patents" (specifically, the section entitled "Treaties and Foreign Patents") for more information on timeframes and deadlines for filing foreign patent applications. The guide is available either by contacting the USPTO Contact Center at 800-786-9199, or it can be viewed on the USPTO website at http://www.uspto.gov/web/offices/pac/doc/general/index.html.

For information on preventing theft of your intellectual property (patents, trademarks and copyrights), you may wish to consult the U.S. Government website, http://www.stopfakes.gov. Part of a Department of Commerce initiative, this website includes self-help "toolkits" giving innovators guidance on how to protect intellectual property in specific countries such as China, Korea and Mexico. For questions regarding patent enforcement issues, applicants may call the U.S. Government hotline at 1-866-999-HALT (1-866-999-4158).

LICENSE FOR FOREIGN FILING UNDER

Title 35, United States Code, Section 184

Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Bureau of Industry and Security, Department of Commerce (15 CFR parts 730-774); the Office of Foreign AssetsControl, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

U.S. PTO
11/980185
10/30/2007

103007 20427 U.S. PTO

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
P.O. Box 1450, Alexandria VA 22313

UTILITY PATENT APPLICATION

SIR:

Please file the following enclosed patent application papers:

Inventors: Lakshmi Arunachalam

Inventor's Addresses: Menlo Park, CA USA

Title: "Method and Apparatus for Enabling Real-Time Transactions on a Network"

(x) Signature by Attorney constituting power of attorney

(x) Specification, Claims, and Abstract: No. of Sheets 41

(x) Inventor's Declaration

(x) Drawings: No. of Sheets 13

(x) Check for $00.00

Including $ 515 Basic Fee - SMALL ENTITY FEE - APPLICANT BY THIS FEE DECLARES ITSELF A SMALL ENTITY
and $ 0 for 0 independent claims above 3 at $105 claim and
$ 25 for 0 claims above 20 at $25 per claim.

Very Respectfully,

Clifford Kraft

Clifford H. Kraft 35,229
Attorney of Record

CORRESPONDENCE ADDRESS
Clifford Kraft
320 Robin Hill Dr.
Naperville, Il. 60540
708 528-9092

Express Mail Label #: EB 579222606 US

Date of Deposit: OCT. 30, 2007

I hereby certify that this paper and fee is being deposited with the United States Postal Service using "Express Mail Post Office To Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to "Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450."

Signed Clifford Kraft Name: CLIFFORD H. KRAFT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	)
	)
Inventor: Lakshmi Arunachalam	)
	)
Serial No.:	)
	)
Filing Date: Oct. 29, 2007	)
	)
Title: METHOD AND APPARATUS FOR ENABLING REAL-TIME TRANSACTIONS ON A NETWORK	)
	)

SUGGESTED RESTRICTION REQUIREMENT (SRR)

Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313=1450

Honorable Commissioner:

Kindly accept this Suggested Restriction Requirement under 37 C.F.R. §1.142(c) as amended by the rules going into effect Nov. 1, 2007.

The applicant respectfully suggests the following restriction:

Claim Group I - Claims 1- 17 drawn to a method for delivering complete wireless transactional services over the World Wide Web.

Claim Group II - Claims 18-24 drawn to an employee-accessible web service network portal .

Claim Group III – Claims 25-32 drawn to an exchange component of a web-based transactional service.

Claim Group IV – Claims 33-39 drawn to a multi-media network object routing service .

Claim Group V – Claims 40-46 drawn to a web service transaction system for allowing N-Way transactions.

Claim Group VI – Claims 47-52 drawn to a cooperative multiple merchant web service system.

Claim Group VII – Claims 53-59 drawn to a method for providing an enhanced value chain between web merchants.

Claim Group VIII – Claims 60-70 drawn to a value added network switch.

Claim Group IX – Claims 72-85 drawn to a method for performing a transaction over a digital network.

Claim Group X – Claims 86-92 drawn to a method for managing services on a network.

Claim Group XI – Claims 93-104 drawn to a method for configuring the context of a service on the World Wide Web.

Claim Group XII – Claims 105-110 drawn to distributed online service information bases.

Respectfully Submitted

Clifford Kraft

Clifford H. Kraft
Reg. No. 35,229
Attorney of Record

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:)

Inventor: Lakshmi Arunachalam)

Serial No.:)

Filing Date: Oct. 30, 2007)

Title: METHOD AND APPARATUS FOR ENABLING REAL-TIME TRANSACTIONS ON A NETWORK)

IDENTIFICATION OF OTHER APPLICATIONS

Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

Honorable Commissioner:

This is an identification of other applications with an inventor in common and a claimed filing or priority date within two months of the claimed filing or priority date of this application as required by 37 C.F.R. §1.78(f)(1).

IDENTIFICATION

This application has a common inventor Lakshmi Arunachalam and a common priority date of Nov. 13, 1995 with the following applications:

08/700,726 now patent no. 5,778,178

08/879,958 now patent no. 5,987,500

09/296,207 now patent no. 6,212,556

09/792,323 co-pending (Notice of allowance issued)

09/863,704 co-pending

Respectfully Submitted

Clifford Kraft

Clifford H. Kraft
Reg. No. 35,229
Attorney of Record

Method and apparatus for enabling real-time bi-directional transactions on a network

This is a continuation-in-part of co-pending application number 09/792,323 filed Feb. 23, 2001 which was a continuation-in-part of application 08/879,958 filed June 20, 1997, now U.S. Patent number 5,987,500, which was a divisional of application number 08/700,726 filed Aug. 5, 1996, now U.S. Patent number 5,778,178, which was related to and claimed priority from provisional application number 60/006,634 filed Nov. 13, 1995. Applications 09/792,323, 08,879,958, 08/700,726 and 60/006,634 are hereby incorporated by reference.

In addition related applications 09/863,704 filed May 23, 2001, 09/296,207 filed April 21, 1999 and provisional application 60/206,422 filed May 23, 2000 are also all hereby incorporated by reference.

The text of this application is substantially similar to that of application 08/700,726, now U.S. Patent 5,778,178.

BACKGROUND

FIELD OF THE INVENTION

The present invention relates to the area of Internet communications. Specifically, the present invention relates to a method and apparatus for configurable value-added network switching and object routing.

BACKGROUND OF THE INVENTION

With the Internet and the World Wide Web ("the Web") evolving rapidly as a viable consumer medium for electronic commerce, new on-line services are emerging to fill the needs of on-line users. An Internet user today can browse on the Web via the use of a Web browser. Web browsers are software interfaces that run on Web clients to allow access to Web servers via a simple user interface. A Web user's capabilities today from a Web browser are, however, extremely limited. The user can perform one-way, browse-only interactions. Additionally, the user has limited "deferred" transactional capabilities, namely electronic mail (e-mail) capabilities. E-mail capabilities are referred to as "deferred transactions" because the consumer's request is not processed until the e-mail is received, read, and the person or system reading the e-mail executes the transaction. This transaction is thus not performed in real-time.

FIG. 1A illustrates typical user interactions on the Web today. User 100 sends out a request from Web browser 102 in the form of a universal resource locator (URL) 101 in the following manner: http://www.car.com. URL 101 is processed by Web browser 102 that determines the URL corresponds to car dealer Web page 105, on car dealer Web server 104. Web browser 102 then establishes browse link 103 to car dealer Web page 105. User 100 can browse Web page 105 and select "hot links" to jump to other locations in Web page 105, or to move to other Web pages on the Web. This interaction is typically a browse-only interaction. Under limited circumstances, the user may be able to fill out a form on car dealer Web page 105, and e-mail the form to car dealer Web server 104. This interaction is still strictly a one-way browse mode communications link, with the e-mail providing limited, deferred transactional capabilities.

Under limited circumstances, a user may have access to two-way services on the Web via Common Gateway Interface (CGI) applications. CGI is a standard interface for running external programs on a Web server. It allows Web servers to create documents dynamically when the server receives a request from the Web browser. When the Web server receives a request for a document, the Web server dynamically executes the appropriate CGI script and transmits the output of the execution back to the requesting Web browser. This interaction can thus be termed a "two-way" transaction. It is a severely limited transaction, however, because each CGI application is customized for a particular type of application or service.

For example, as illustrated in FIG. 1B, user 100 may access bank 150's Web server and attempt to perform transactions on checking account 152 and to make a payment on loan account 154. In order for user 100 to access checking account 152 and loan account 154 on the Web, CGI application scripts must be created for each account, as illustrated in FIG. 1B. The bank thus has to create individual scripts for each of its services to offer users access to these services. User 100 can then interact in a limited fashion with these individual applications. Creating and managing individual CGI scripts for each service is not a viable solution for merchants with a large number of services.

As the Web expands and electronic commerce becomes more desirable, the need increases for robust, real-time, bi-directional transactional capabilities on the Web. A true real-time, bi-directional transaction would allow a user to connect to a variety of services on the Web, and perform real-time transactions on those services. For example, although user 100 can browse car dealer Web page 105 today, the user cannot purchase the car, negotiate a car loan or perform other types of real-time, two-way transactions that he can perform with a live salesperson at the car dealership. Ideally, user 100 in FIG. 1A would be able to access car dealer Web page 105, select specific transactions that he desires to perform, such as purchase a car, and perform the purchase in real-time, with two-way interaction capabilities. CGI applications provide user 100 with a limited ability for two-way interaction with car dealer Web page 105, but due to the lack of interaction and management between the car dealer and the bank, he will not be able to obtain a loan and complete the purchase of the car via a CGI application. The ability to complete robust real-time, two-way transactions is thus not truly available on the Web today.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a method and apparatus for providing real-time, two-way transactional capabilities on the Web. Specifically, one embodiment of the present invention discloses a configurable value-added network switch for enabling real-time transactions on the World Wide Web. The configurable value added network switch comprises means for switching to a transactional application in response to a user specification from a World Wide Web application, means for transmitting a transaction request from the transactional application, and means for processing the transaction request.

According to another aspect of the present invention, a method and apparatus for enabling object routing on the World Wide Web is disclosed. The method for enabling object routing comprises the steps of creating a virtual information store containing information entries and attributes, associating each of the information entries and the attributes with an object identity, and assigning a unique network address to each of the object identities.

Other objects, features and advantages of the present invention will be apparent from the accompanying drawings and from the detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention will be apparent from the accompanying drawings and from the detailed description of the present invention as set forth below.

FIG. 1A is an illustration of a current user's browse capabilities on the Web via a Web browser.

FIG. 1B is an illustration of a current user's capabilities to perform limited transactions on the Web via CGI applications.

FIG. 2 illustrates a typical computer system on which the present invention may be utilized.

FIG. 3 illustrates the Open Systems Interconnection (OSI) Model.

FIG. 4A illustrates conceptually the user value chain as it exists today.

FIG. 4B illustrates one embodiment of the present invention.

FIG. 5A illustrates a user accessing a Web server including one embodiment of the present invention.

FIG. 5B illustrates the exchange component according to one embodiment of the present invention.

FIG. 5C illustrates an example of a point-of-service (POSvc) application list.

FIG. 5D illustrates a user selecting a bank POSvc application from the POSvc application list.

FIG. 5E illustrates a three-way transaction according to one embodiment of the present invention.

FIG. 6A illustrates a value-added network (VAN) switch.

FIG. 6B illustrates the hierarchical addressing tree structure of the networked objects in DOLSIBs.

FIG. 7 illustrates conceptually the layered architecture of a VAN switch.

FIG. 8 is a flow diagram illustrating one embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention relates to a method and apparatus for configurable value-added network switching and object routing and management. "Web browser" as used in the context of the present specification includes conventional Web browsers such as NCSA Mosaic TM. from NCSA and Netscape Mosaic TM. from Netscape TM.. The present invention is independent of the Web browser being utilized and the user can use any Web browser, without modifications to the Web browser. In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent to one of ordinary skill in the art, however, that these specific details need not be used to practice the present invention. In other instances, well-known structures, interfaces and processes have not been shown in detail in order not to unnecessarily obscure the present invention.

FIG. 2 illustrates a typical computer system 200 in which the present invention operates. \The preferred embodiment of the present invention is implemented on an IBM.TM. Personal Computer manufactured by IBM Corporation of Armonk, N.Y. Alternate embodiments may be implemented on a Macintosh TM. computer manufactured by Apple TM. Computer, Incorporated of Cupertino, Calif. It will be apparent to those of ordinary skill in the art that other alternative computer system architectures may also be employed.

In general, such computer systems as illustrated by FIG. 2 comprise a bus 201 for communicating information, a processor 202 coupled with the bus 201 for processing information, main memory 203 coupled with the bus 201 for storing information and instructions for the processor 202, a read-only memory 204 coupled with the bus 201 for storing static information and instructions for the processor 202, a display device 205 coupled with the bus 201 for displaying information for a computer user, an input device 206 coupled with the bus 201 for communicating information and command selections to the processor 202, and a mass storage device 207, such as a magnetic disk and associated disk drive, coupled with the bus 201 for storing information and instructions. A data storage medium 208 containing digital information is configured to operate with mass storage device 207 to allow processor 202 access to the digital information on data storage medium 208 via bus 201.

Processor 202 may be any of a wide variety of general purpose processors or microprocessors such as the Pentium.TM. microprocessor manufactured by Intel.TM. Corporation or the Motorola.TM. 68040 or Power PC.TM. brand microprocessor manufactured by manufactured by Motorola.TM. Corporation. It will be apparent to those of ordinary skill in the art, however, that other varieties of processors may also be used in a particular computer system. Display device 205 may be a liquid crystal device, cathode ray tube (CRT), or other suitable display device. Mass storage device 207 may be a conventional hard disk drive, floppy disk drive, CD-ROM drive, or other magnetic or optical data storage device for reading and writing information stored on a hard disk, a floppy disk, a CD-ROM a magnetic tape, or other magnetic or optical data storage medium. Data storage medium 208 may be a hard disk, a floppy disk, a CD-ROM, a magnetic tape, or other magnetic or optical data storage medium.

In general, processor 202 retrieves processing instructions and data from a data storage medium 208 using mass storage device 207 and downloads this information into random access memory 203 for execution. Processor 202, then executes an instruction stream from random access memory 203 or read-only memory 204. Command selections and information input at input device 206 are used to direct the flow of instructions executed by processor 202. Equivalent input device 206 may also be a pointing device such as a conventional mouse or trackball device. The results of this processing execution are then displayed on display device 205.

The preferred embodiment of the present invention is implemented as a software module, which may be executed on a computer system such as computer system 200 in a conventional manner. Using well known techniques, the application software of the preferred embodiment is stored on data storage medium 208 and subsequently loaded into and executed within computer system 200. Once initiated, the software of the preferred embodiment operates in the manner described below.

FIG. 3 illustrates the Open Systems Interconnection (OSI) reference model. OSI Model 300 is an international standard that provides a common basis for the coordination of standards development, for the purpose of systems interconnection. The present invention is implemented to function as a routing switch within the "application layer" of the OSI model. The model defines seven layers, with each layer communicating with its peer layer in another node through the use of a protocol. Physical layer 301 is the lowest layer, with responsibility to transmit unstructured bits across a link. Data link layer 302 is the next layer above physical layer 301. Data link layer 302 transmits chunks across the link and deals with problems like checksumming to detect data corruption, orderly coordination of the use of shared media and addressing when multiple systems are reachable. Network bridges operate within data link layer 302.

Network layer 303 enables any pair of systems in the network to communicate with each other. Network layer 303 contains hardware units such as routers, that handle routing, packet fragmentation and reassembly of packets. Transport layer 304 establishes a reliable communication stream between a pair of systems, dealing with errors such as lost packets, duplicate packets, packet reordering and fragmentation. Session layer 305 offers services above the simple communication stream provided by transport layer 304. These services include dialog control and chaining. Presentation layer 306 provides a means by which OSI compliant applications can agree on representations for data. Finally, application layer 307 includes services such as file transfer, access and management services (FTAM), electronic mail and virtual terminal (VT) services. Application layer 307 provides a means for application programs to access the OSI environment. As described above, the present invention is implemented to function as a routing switch in application layer 307. Application layer routing creates an open channel for the management, and the selective flow of data from remote databases on a network.

A. OVERVIEW

FIG. 4A illustrates conceptually the user value chain as it exists today. The user value chain in FIG. 4A depicts the types of transactions that are performed today, and the channels through which the transactions are performed. A "transaction" for the purposes of the present invention includes any type of commercial or other type of interaction that a user may want to perform. Examples of transactions include a deposit into a bank account, a request for a loan from a bank, a purchase of a car from a car dealership or a purchase of a car with financing from a bank. A large variety of other transactions are also possible.

A typical user transaction today may involve user 100 walking into a bank or driving up to a teller machine, and interacting with a live bank teller, or automated teller machine (ATM) software applications. Alternatively, user 100 can perform the same transaction by using a personal computer (PC), activating application software on his PC to access his bank account, and dialing into the bank via a modem line. If user 100 is a Web user, however, there is no current mechanism for performing a robust, real-time transaction with the bank, as illustrated in FIG. 4A. CGI scripts provide only limited two-way capabilities, as described above. Thus, due to this lack of a robust mechanism by which real-time Web transactions can be performed, the bank is unable to be a true "Web merchant," namely a merchant capable of providing complete transactional services on the Web.

According to one embodiment of the present invention, as illustrated in FIG. 4B, each merchant that desires to be a Web merchant can provide real-time transactional capabilities to users who desire to access the merchants' services via the Web. This embodiment includes a service network running on top of a facilities network, namely the Internet, the Web or e-mail networks. For the purposes of this application, users are described as utilizing PC's to access the Web via Web server "switching" sites. (Switching is described in more detail below). Users may also utilize other personal devices such as network computers or cellular devices to access the merchants' services via appropriate switching sites. These switching sites include non-Web network computer sites and cellular provider sites. Five components interact to provide this service network functionality, namely an exchange, an operator agent, a management agent, a management manager and a graphical user interface. All five components are described in more detail below.

As illustrated in FIG. 5A, user 100 accesses Web server 104. Having accessed Web server 104, user 100 can decide that he desires to perform real-time transactions. When Web server 104 receives user 100's indication that he desires to perform real-time transactions, the request is handed over to an exchange component. Thus, from Web page 105, for example, user 100 can select button 500, entitled "Transactions" and Web server 104 hands user 100's request over to the exchange component. The button and the title can be replaced by any mechanism that can instruct a Web server to hand over the consumer's request to the exchange component.

FIG. 5B illustrates exchange 501. Exchange 501 comprises Web page 505 and point-of-service (POSvc) applications 510. Exchange 501 also conceptually includes a switching component and an object routing component (described in more detail below). POSvc applications 510 are transactional applications, namely applications that are designed to incorporate and take advantage of the capabilities provided by the present invention. Although exchange 501 is depicted as residing on Web server 104, the exchange can also reside on a separate computer system that resides on the Internet and has an Internet address. Exchange 501 may also include operator agent 503 that interacts with a management manager (described in more detail below). Exchange 501 creates and allows for the management (or distributed control) of a service network, operating within the boundaries of an IP-based facilities network. Thus, exchange 501 and a management agent component, described in more detail below, under the headings "VAN Switch and Object Routing," together perform the switching, object routing, application and service management functions according to one embodiment of the present invention.

Exchange 501 processes the consumer's request and displays an exchange Web page 505 that includes a list of POSvc applications 510 accessible by exchange 501. A POSvc application is an application that can execute the type of transaction that the user may be interested in performing. The POSvc list is displayed via the graphical user interface component. One embodiment of the present invention supports HyperText Markup Language as the graphical user interface component. Virtual Reality Markup Language and Java.TM. are also supported by this embodiment. A variety of other graphical user interface standards can also be utilized to implement the graphical user interface.

An example of a POSvc application list is illustrated in FIG. 5C. User 100 can thus select from POSvc applications Bank 510(1), Car Dealer 510(2) or Pizzeria 510(3). Numerous other POSvc applications can also be included in this selection. If user 100 desires to perform a number of banking transactions, and selects the Bank application, a Bank POSvc application will be activated and presented to user 100, as illustrated in FIG. 5D. For the purposes of illustration, exchange 501 in FIG. 5D is shown as running on a different computer system (Web server 104) from the computer systems of the Web merchants running POSvc applications (computer system 200). Exchange 501 may, however, also be on the same computer system as one or more of the computer systems of the Web merchants.

Once Bank POSvc application 510 has been activated, user 100 will be able to connect to Bank services and utilize the application to perform banking transactions, thus accessing data from a host or data repository 575 in the Bank "Back Office." The Bank Back Office comprises legacy databases and other data repositories that are utilized by the Bank to store its data. This connection between user 100 and Bank services is managed by exchange 501. As illustrated in FIG. 5D, once the connection is made between Bank POSvc application 510(1), for example, and Bank services, an operator agent on Web server 104 may be activated to ensure the availability of distributed functions and capabilities.

Each Web merchant may choose the types of services that it would like to offer its clients. In this example, if Bank decided to include in their POSvc application access to checking and savings accounts, user 100 will be able to perform real-time transactions against his checking and savings accounts. Thus, if user 100 moves $500 from his checking account into his savings account, the transaction will be performed in real-time, in the same manner the transaction would have been performed by a live teller at the bank or an ATM machine. Therefore, unlike his prior access to his account, user 100 now has the capability to do more than browse his bank account. The ability to perform these types of robust, real-time transactions from a Web client is a significant aspect of the present invention.

Bank can also decide to provide other types of services in POSvc application 510(1). For example, Bank may agree with Car dealership to allow Bank customers to purchase a car from that dealer, request a car loan from Bank, and have the entire transaction performed on the Web, as illustrated in FIG. 5E. In this instance, the transactions are not merely two-way, between the user and Bank, but three-way, amongst the consumer, Bank and Car dealership. According to one aspect of the present invention, this three-way transaction can be expanded to n-way transactions, where n represents a predetermined number of merchants or other service providers who have agreed to cooperate to provide services to users. The present invention therefore allows for "any-to-any" communication and transactions on the Web, thus facilitating a large, flexible variety of robust, real-time transactions on the Web.

Finally, Bank may also decide to provide intra-merchant or intra-bank services, together with the inter-merchant services described above. For example, if Bank creates a POSvc application for use by the Bank Payroll department, Bank may provide its own employees with a means for submitting timecards for payroll processing by the Bank's Human Resources (HR) Department. An employee selects the Bank HR POSvc application, and submits his timecard. The employee's timecard is processed by accessing the employee's payroll information, stored in the Bank's Back Office. The transaction is thus processed in real-time, and the employee receives his paycheck immediately.

B. VAN SWITCHING AND OBJECT ROUTING

As described above, exchange 501 and management agent 601, illustrated in FIG. 6A, together constitute a value-added network (VAN) switch. These two elements may take on different roles as necessary, including peer-to-peer, client-server or master-slave roles. Management manager 603 is illustrated as residing on a separate computer system on the Internet. Management manager 603 can, however, also reside on the same machine as exchange 501. Management manager 603 interacts with the operator agent 503 residing on exchange 501.

VAN switch 520 provides multi-protocol object routing, depending upon the specific VAN services chosen. This multi-protocol object routing is provided via a proprietary protocol, TransWeb.TM. Management Protocol (TMP). TMP incorporates the same security features as the traditional Simple Network Management Protocol, SNMP. It also allows for the integration of other traditional security mechanisms, including RSA security mechanisms.

One embodiment of the present invention utilizes TMP and distributed on-line service information bases (DOLSIBs) to perform object routing. Alternatively, TMP can incorporate s-HTTP, Java.TM., the WinSock API or ORB with DOLSIBs to perform object routing. DOLSIBs are virtual information stores optimized for networking. All information entries and attributes in a DOLSIB virtual information store are associated with a networked object identity. The networked object identity identifies the information entries and attributes in the DOLSIB as individual networked objects, and each networked object is assigned an Internet address. The Internet address is assigned based on the IP address of the node at which the networked object resides.

For example, in FIG. 5A, Web server 104 is a node on the Internet, with an IP address. All networked object associated with Web server 104 will therefore be assigned an Internet address based on the Web server 104's IP address. These networked objects thus "branch" from the node, creating a hierarchical tree structure. The Internet address for each networked object in the tree essentially establishes the individual object as an "IP-reachable" or accessible node on the Internet. TMP utilizes this Internet address to uniquely identify and access the object from the DOLSIB. FIG. 6B illustrates an example of this hierarchical addressing tree structure.

Each object in the DOLSIB has a name, a syntax and an encoding. The name is an administratively assigned object ID specifying an object type. The object type together with the object instance serves to uniquely identify a specific instantiation of the object. For example, if object 610 is information about models of cars, then one instance of that object would provide user 100 with information about a specific model of the car while another instance would provide information about a different model of the car. The syntax of an object type defines the abstract data structure corresponding to that object type. Encoding of objects defines how the object is represented by the object type syntax while being transmitted over the network.

C. <u>MANAGEMENT AND ADMINISTRATION</u>

As described above, exchange 501 and management agent 601 together constitute a VAN switch. FIG. 7 illustrates conceptually the layered architecture of VAN switch 520. Specifically, boundary service 701 provides the interfaces between VAN switch 520, the Internet and the Web, and multi-media end user devices such as PCs, televisions or telephones. Boundary service 701 also provides the interface to the on-line service provider. A user can connect to a local application, namely one accessible via a local VAN switch, or be routed or "switched" to an application accessible via a remote VAN switch.

Switching service 702 is an OSI application layer switch. Switching service 702 thus represents the core of the VAN switch. It performs a number of tasks including the routing of user connections to remote VAN switches, described in the paragraph above, multiplexing and prioritization of requests, and flow control. Switching service 702 also facilitates open systems' connectivity with both the Internet (a public switched network) and private networks including back office networks, such as banking networks. Interconnected application layer switches form the application network backbone. These switches are one significant aspect of the present invention.

Management service 703 contains tools such as Information Management Services (IMS) and application Network Management Services (NMS). These tools are used by the end users to manage network resources, including VAN switches. Management service 703 also provides applications that perform Operations, Administration, Maintenance & Provisioning (OAM&P) functions. These OAM&P functions include security management, fault management, configuration management, performance management and billing management. Providing OAM&P functions for applications in this manner is another significant aspect of the present invention.

`Finally, application service 704 contains application programs that deliver customer services. Application service 704 includes POSvc applications such as Bank POSvc described above, and illustrated in FIG. 6A. Other examples of VAN services include multi-media messaging, archival/retrieval management, directory services, data staging, conferencing, financial services, home banking, risk management and a variety of other vertical services. Each VAN service is designed to meet a particular set of requirements related to performance, reliability, maintenance and ability to handle expected traffic volume. Depending on the type of service, the characteristics of the network elements will differ. VAN service 704 provides a number of functions including communications services for both management and end users of the network and control for the user over the user's environment.

FIG. 8 is a flow diagram illustrating one embodiment of the present invention. A user connects to a Web server running an exchange component in step 802. In step 804, the user issues a request for a transactional application, and the web server hands off the request to an exchange in step 806. The exchange activates a graphical user interface to present user with a list of POSvc application options in step 808. In step 810, the user makes a selection from the POSvc application list. In step 812, the switching component in the exchange switches the user to the selected POSvc application, and in step 814, the object routing component executes the user's request. Data is retrieved from the appropriate data repository via TMP in step 816, and finally, the user may optionally continue the transaction in step 818 or end the transaction.

Thus, a configurable value-added network switching and object routing method and apparatus is disclosed. These specific arrangements and methods described herein are merely illustrative of the principles of the present invention. Numerous modifications in form and detail may be made by those of ordinary skill in the art without departing from the scope of the present invention. Although this invention has been shown in relation to a particular preferred embodiment, it should not be considered so limited. Rather, the present invention is limited only by the scope of the appended claims.

CLAIMS

1. A method for delivering complete wireless transactional services over the World Wide Web comprising the steps of:

receiving a transactional request from a wireless user for access to media content;

handing said transactional request to an exchange component, said exchange component providing said wireless user with a choice of currently available media content services accessible by said exchange component;

receiving a selection of a particular accessible media content service from said wireless user;

providing a choice of available media content from said particular media content service to said wireless user;

receiving a request from said wireless user for particular media content.

providing said particular wireless media content in real time to said wireless user.

2. The method for delivering complete wireless transactional services over the World Wide Web of claim 1 wherein said particular wireless media content includes video.

3. The method for delivering complete wireless transactional services over the World Wide Web of claim 1 wherein said particular wireless media content includes audio.

4. The method for delivering complete wireless transactional services over the World Wide Web of claim 1 wherein said particular wireless media content includes web advertising.

5. The method for delivering complete wireless transactional services over the World Wide Web of claim 1 wherein said particular wireless media content includes buying or selling.

6. The method for delivering complete wireless transactional services over the World Wide Web of claim 1 wherein said particular wireless media content is multi-media.

7. The method for delivering complete wireless transactional services over the World Wide Web of claim 1 wherein the step of providing said particular wireless media content is performed through a switching or exchange component.

8. The method for delivering complete wireless transactional services over the World Wide Web of claim 7 wherein said switching or exchange component provides a plurality of vertical services.

9. The method for delivering complete wireless transactional services over the World Wide Web of claim 8 wherein said vertical services are chosen from the group consisting of messaging, archival retrieval, directory services, data staging and financial services.

10. A system for delivering complete wireless transactional services over the World Wide Web comprising:

a management component capable of communicating with a wireless user, said management component receiving a request from a wireless user for wireless media content services;

an exchange component supplying said wireless user with a choice of available wireless media services, wherein said exchange component receives a choice by said wireless user relating to a particular wireless media service;

a switching component providing information transfer between said particular wireless media service and said wireless user by which said wireless user may choose and receive particular wireless media content.

11. The system for delivering complete wireless transactional services over the World Wide Web of claim 10 wherein said particular wireless media content includes video.

12. The system for delivering complete wireless transactional services over the World Wide Web of claim 10 wherein said particular wireless media content includes audio.

13. The system for delivering complete wireless transactional services over the World Wide Web of claim 10 wherein said particular wireless media content includes web advertising.

14. The system for delivering complete wireless transactional services over the World Wide Web of claim 10 wherein said particular wireless media content is multi-media.

15. The system for delivering complete wireless transactional services over the World Wide Web of claim 10 wherein said particular wireless media content includes web buying and selling.

16. The system for delivering complete wireless transactional services over the World Wide Web of claim 10 wherein said switching component provides a plurality of vertical services.

17. The system for delivering complete wireless transactional services over the World Wide Web of claim 16 wherein said vertical services are chosen from the group consisting of messaging, archival retrieval, directory services, data staging and financial services.

18. An employee-accessible web service network portal operated by a business entity comprising:

a point of service application provided by a particular sub-entity related to said business entity;

a second application provided by a different sub-entity also related to said business entity;

a portal allowing an employee access to said point of service application, said portal also allowing said employee to transfer information from said second application to said point of service application.

19. The employee-accessible web service network portal of claim 18 wherein said particular sub-entity is a payroll department.

20. The employee-accessible web service network portal of claim 18 wherein said different sub-entity is a human resources department.

21. The employee-accessible web service network portal of claim 18 wherein funds can be transferred by said point of service application to benefit said employee.

22 The employee-accessible web service network portal of claim 18 wherein said portal allows said employee to transfer information between other application programs provided by other sub-entities related to said business entity.

23. The employee-accessible web service network portal of claim 18 wherein one of said point of service application allows access to the group of services consisting of 401K plans, expense reports, time cards, payroll, travel, vacation and commissions.

24. The employee-accessible web service network portal of claim 18 further comprising a plurality of point of service applications.

25. An exchange component of a web-based transactional service comprising:

a plurality of application components;

a switching component;

an object routing component;

a web page component;

wherein said web page component provides a web page to a user that allows said user to select a particular transactional service, said switching component switches information between said user and an application component related to said particular transactional service, and said object routing component routes media content objects between said particular transactional service and said user.

26. The exchange component of a web-based transactional service of claim 25 wherein said switching component is a value added network switch.

27. The exchange component of a web-based transactional service of claim 25 wherein some of said application programs are point of service applications.

28. The exchange component of a web-based transactional service of claim 25 wherein said exchange component receives a handoff message from a server when a user requests said particular transactional service.

29. The exchange component of a web-based transactional service of claim 28 wherein said server is remote from said exchange component.

30. The exchange component of a web-based transactional service of claim 25 wherein said web page component provides said user with a graphical user interface containing a list of available transactional services.

31. The exchange component of a web-based transactional service of claim 30 wherein said web page component allows said user to choose a transactional service from said list.

32. The exchange component of a web-based transactional service of claim 31 wherein said switching component switches said user to one of said application components based upon a choice from said list.

33. A multi-media network object routing service comprising:

a virtual information store located on one or more networked computers containing a plurality of information entries and a plurality of object attributes, wherein each of said information entries and each of said object attributes is associated with a particular multi-media web service network object, said multi-media web service network object having both a network object identity and a unique network address.

34. The multi-media web service network object routing service of claim 33 wherein said unique network address is an IP address.

35. The multi-media web service network object routing service of claim 33 wherein multi-media web service network object resides on a node having an IP address.

36. The multi-media web service network object routing service of claim 35 wherein said unique network address is based on the IP address of said node.

37. The multi-media web service network object routing service of claim 35 wherein said node is a web server having a particular IP address.

38. The multi-media web service network object routing service of claim 37 wherein said unique network address is related to said particular IP address of said web server.

39. The multi-media web service network object routing service of claim 37 wherein said unique network address is related hierarchically to said particular IP address of said web server.

40. A web service transaction system for allowing N-Way transactions comprising:

a web-based application accessible by N web participants, where N is an integer greater than 1, each of said web participants providing a service, and wherein said web-based application allows transfer of information between members of said N web participants;

a user interface to said web-based application, wherein a user can access a service from at least one of said N web participants;

and wherein said web-based application notifies at least one of said web participants when the user accesses a service from another of said web participants.

41. The web service transaction system for allowing N-Way transactions of claim 40 wherein at least one of said N web participants is a merchant.

42. The web service transaction system for allowing N-Way transactions of claim 40 wherein at least one of said services includes advertising.

43. The web service transaction system for allowing N-Way transactions of claim 40 wherein said web-based application also sends advertising to said user.

44. The web service transaction system for allowing N-Way transactions of claim 43 wherein said advertising originates at one of said N web participants.

45. The web service transaction system for allowing N-Way transactions of claim 40 wherein at least one of said N participants exchanges multi-media information with said user in real time.

46. The web service transaction system for allowing N-Way transactions of claim 40 wherein N is an integer greater than 2.

47. A cooperative multiple merchant web service system comprising:

at least one point of service application accessible by a plurality of web merchants, each of said web merchants providing goods or services, said point of service application allowing transfer of information between said web merchants;

a user interface to said point of service application, wherein said user can access at least some of said goods or services, and wherein access by said user to one of said merchant's goods or services is communicated to at least one other of said merchants.

48. The cooperative multiple merchant web service system of claim 47 wherein a particular one of said web merchants provides the service of maintaining a record of said user's accesses.

49. The cooperative multiple merchant web service system of claim 48 wherein said particular one of said merchants allows said user to provide funds for at least some of said accesses.

50. The cooperative multiple merchant web service system of claim 47 wherein one of said merchants is a financial institution.

51. The cooperative multiple merchant web service system of claim 47 wherein at least one of said merchants provides fungible goods.

52. The cooperative multiple merchant web service system of claim 47 wherein at least one of said merchants provides exchanges multi-media information with said user.

53. A method for providing an enhanced value chain between web merchants and users comprising the steps of:

providing a service network running on the internet upon which a plurality of web merchants provide real-time point of service transactional capabilities;

providing at least one web site where a user can access said service network;

providing an exchange component that interacts with said web site, wherein said exchange component provides said user with information relating to available point of service applications;

allowing the user to choose a particular point of service application and to interact with that particular point of service application to complete a real-time transaction over the Web.

54. The method for providing an enhanced value chain between web merchants and users of claim 53 wherein said exchange component communicates with a switching component.

55. The method for providing an enhanced value chain between web merchants and users of claim 54 wherein said switching component routes information between said user and said particular point of service application.

56. The method for providing an enhanced value chain between web merchants and users of claim 53 wherein users are provided with a list of available point of service applications.

57. The method for providing an enhanced value chain between web merchants and users of claim 53 wherein said exchange component communicates with an object routing component.

58. The method for providing an enhanced value chain between web merchants and users of claim 57 wherein said object routing component allows completion of said real-time transaction.

59. The method for providing an enhanced value chain between web merchants and users of claim 53 wherein said user may be a supplier, partner, distributor or value-added resellers.

60. A value added web service network switch comprising, in combination:

a switching service component that routes information between users and sources of customer services;

a boundary service component that provides an interface to an enterprise network and to a user connected to a facilities network;

an application service component containing application programs that deliver said customer services to users from said enterprise network.

61. The value added web service network switch of claim 60 wherein said facilities network is the Web.

62. The value added web service network switch of claim 60 wherein said application service component delivers services from the group consisting of multi-media messaging, archival/retrieval management, directory services, data staging, conferencing, financial services, home banking and risk management.

63. The value added web service network switch of claim 60 further comprising a management component wherein said management component provides operations, administration, maintenance or provisioning functions.

64. The value added web service network switch of claim 60 further comprising a management component that provides information management services, or application network management services or a control center .

65. The value added web service network switch of claim 60 wherein said boundary service component interfaces to another value-added network switch.

66. The value added web service network switch of claim 60 wherein said boundary service component interfaces to personal computers, televisions or telephones.

67. The value added web service network switch of claim 60 wherein said boundary service interfaces to a private network.

68. The value added web service network switch of claim 60 wherein said switching service component performs multiplexing or prioritization of requests or flow control of selective flow of data or multi-media information on one or more channels.

69. The value added web service network switch of claim 60 wherein said boundary service component interfaces with a plurality of other value added web service network switches.

70. The value added web service network switch of claim 60 wherein said application service component provides billing over the Web.

71. The value added web service network switch of claim 60 wherein said application service component provides security management over the Web.

72. A method for performing a real time transaction over a digital network, the method comprising:

providing a web page for display on a computer system, wherein a user input device is coupled to the computer system;

providing a point of service application as a selection within the web page, wherein the point of service application provides access to both a checking and savings account;

accepting a first signal from the user input device to select the point of service application;

accepting subsequent signals from the user input device; and

transferring, in real-time and in response to the subsequent signals, funds from the checking account to the savings account.

73. The method of claim 72, further comprising:

using a web service exchange to complete the transfer of funds.

74. The method of claim 72, further comprising:

using a management agent to complete the transfer of funds.

75. The method of claim 72, further comprising:

using object routing to complete the transfer of funds.

76. The method of claim 75, wherein the object routing includes:

using distributed on-line service information bases.

77. The method of claim 72, further comprising:

using a virtual information store to complete the transfer of funds.

78. The method of claim 77, wherein the virtual information store includes a web service networked object.

79. The method of claim 78, wherein the networked object includes a networked object identity.

80. The method of claim 78, wherein each networked object is assigned an Internet address.

81. The method of claim 80, wherein the Internet address is assigned based on the node at which the networked object resides.

82. The method of claim 81, wherein a hierarchical addressing tree structure is used to assign the Internet address.

83. The method of claim 72 wherein said transaction is requesting a loan from a lender.

84. The method of claim 72 wherein said transaction is purchasing a vehicle with financing from a bank.

85. The method of claim 72 wherein said transaction is accessing an account.

86. A method for managing services on a web service network in order to facilitate a transaction, the method comprising:

providing a management service to a user;

accepting signals from a user input device operated by the user to control the management service; and

using the information management service to process an object identity, wherein the object identity represents a networked object.

87. The method of claim 86, wherein the management service includes an information management service.

88. The method of claim 86, wherein the management service includes a web service network management service.

89. The method of claim 86, wherein the management service includes an operations, administration, maintenance and provisioning function on the Web.

90. The method of claim 89, wherein the function includes security management on the Web.

91. The method of claim 89, wherein the function includes fault management on the Web.

92. The method of claim 89, wherein the function includes configuration management on the Web.

93. A method for configuring the context of a service on the World Wide Web comprising the steps of:

providing a user interface that defines objects, information entries and attributes;

storing said information entries and attributes in a virtual information store;

configuring an instance of a process model;

associating said process model with an object having an object identity that includes said information entries and attributes on the World Wide Web;

assigning a unique network address to said object identity;

performing object routing of said object on the World Wide Web;

creating a web-based virtual accelerator as a web service;

creating a grid network on the World Wide Web, said grid network having a plurality of grains;

instantiating a context on the World Wide Web;

associating a grain of said grid network to said context on the World Wide Web; using said web-based virtual accelerator to access said context and said object on the World Wide Web.

94. The method of claim 93, wherein the process model is an organizational process model.

95. The method of claim 93, wherein the process model is a business process model.

96 The method of claim 93, wherein the process model is context-driven.

97. The method of claim 93, wherein the process model relates to a group of geographies, markets, industries, departments, organizations, enterprises, web services, classifications and types.

98. The method of claim 93, wherein the grains relate to groups of geographies, markets, industries, departments, organizations, enterprises, web services, classifications and types.

99. The method of claim 93, wherein the grid network includes grains of instances of a group of process elements, context, web applications, geographies, markets, industries, departments, organizations, enterprises, web services, classifications and types.

100. The method of claim 93, wherein the context is distinct from content on the World Wide Web.

101. The method of claim 93, wherein the context is Business Process Automation.

102. The method of claim 93, wherein the user interface provides a common command view of web services.

103. The method of claim 93, wherein the user interface includes a control center for the grid network of context elements.

104. The method of claim 93, wherein the grid network includes open access to unified communication using object routing.

105. A distributed online service information base on the World Wide Web comprising:

a virtual information store containing a plurality of information entries and attributes, wherein said information entries and attributes are associated with a particular networked object having an object identity.

106. The distributed online service information base on the World Wide Web of claim 105 wherein each networked object has a network address.

107. The distributed online service information base on the World Wide Web of claim 106 wherein said network address is an internet address.

108. The distributed online service information base on the World Wide Web of claim 107 wherein said internet address is an IP address.

109. The distributed online service information base on the World Wide Web of claim 107 wherein said internet address is associated with the node where said object is stored.

110. The distributed online service information base on the World Wide Web of claim 105 wherein said object has a name and an object type.

ABSTRACT

The present invention provides a method and apparatus for providing real-time, two-way transactional capabilities on the Web. Specifically, one embodiment of the present invention discloses a method for enabling object routing, the method comprising the steps of creating a virtual information store containing information entries and attributes associating each of the information entries and the attributes with an object identity, and assigning a unique network address to each of the object identities. A method is also disclosed for enabling service management of the value-added network service, to perform OAM&P functions on the services network

DECLARATION FOR UTILITY PATENT

As a below-named inventor, I hereby declare the my residence, post office address, and citizenship are as stated below next to my name and that I believe that I am the original, first, and sole inventor [if only one name is listed below] or an original, first and joint inventor [if plural names are listed below] of the subject matter which is claimed and for which a patent is sought on the invention, the specification of which is attached hereto and which has the following title:

TITLE: "Method and apparatus for enabling real-time bi-directional transactions on a network"

I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment specifically referred to in the oath or declaration. I acknowledge a duty to disclose information which is material to the examination of this application in accordance with 37 C.F.R. 1.56(a). In addition, if this is a continuation-in-part application under 35 U.S.C. 120, I acknowledge the duty to disclose all information known to be material to patentability as defined in 37 C.F.R. 1.56 which becomes available between the filing date of the prior application and the filing date of the continuation-in-part.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Title 18, United States Code, Section 1001, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

With this document, I also give Power of Attorney to Clifford H. Kraft, 320 Robin Hill Dr., Naperville, IL 60540 Reg. No. 35,229 to act for me in this matter before the United States Patent and Trademark Office.

CORRESPONDENCE ADDRESS:

Clifford H. Kraft
320 Robin Hill Dr.
Naperville, IL 60540

Signature 1st Inventor: Lakshmi Arunachalam
Print Name: Lakshmi Arunachalam Date: October 25, 2007
Residence: Menlo Park, CA Citizen of: US
Address: 222 Stanford Avenue, Menlo Park, CA 94025
Telephone: 650 854-3393

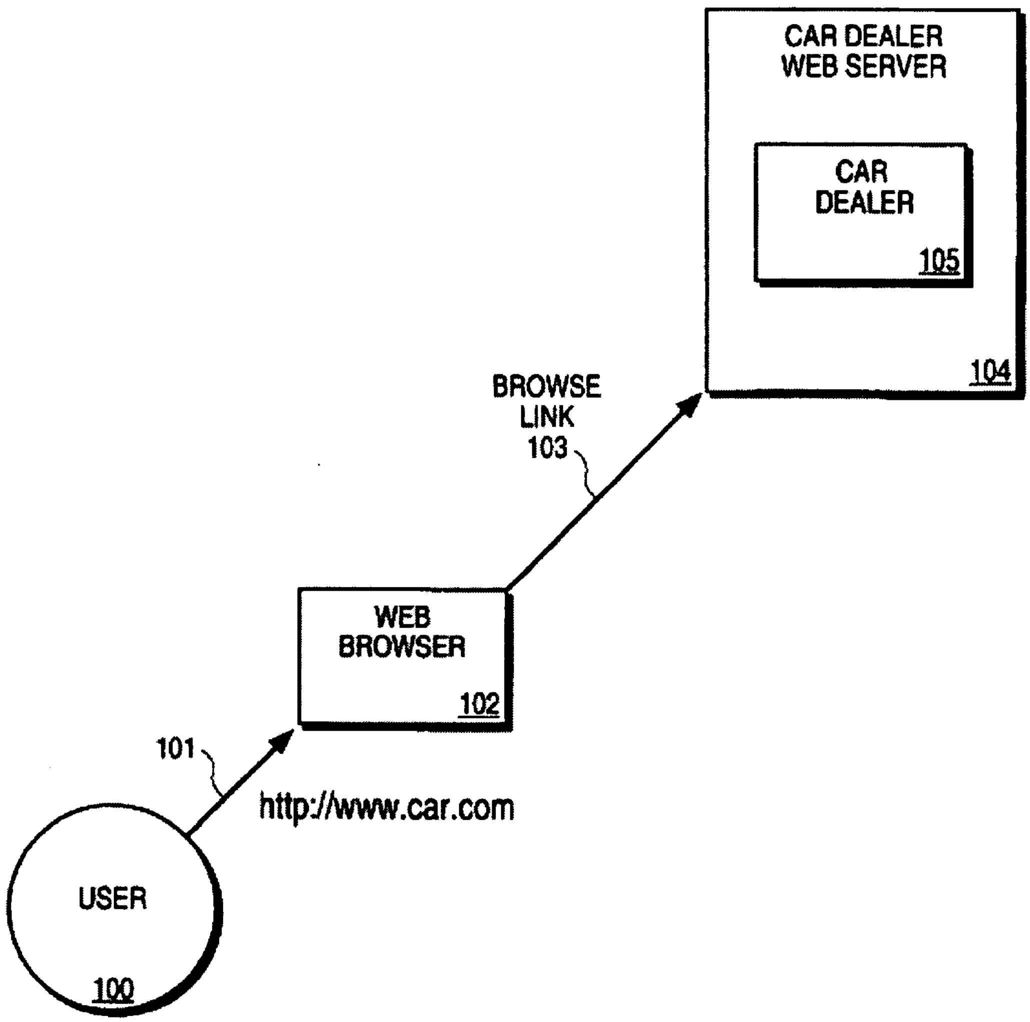


FIG. 1A (PRIOR ART)

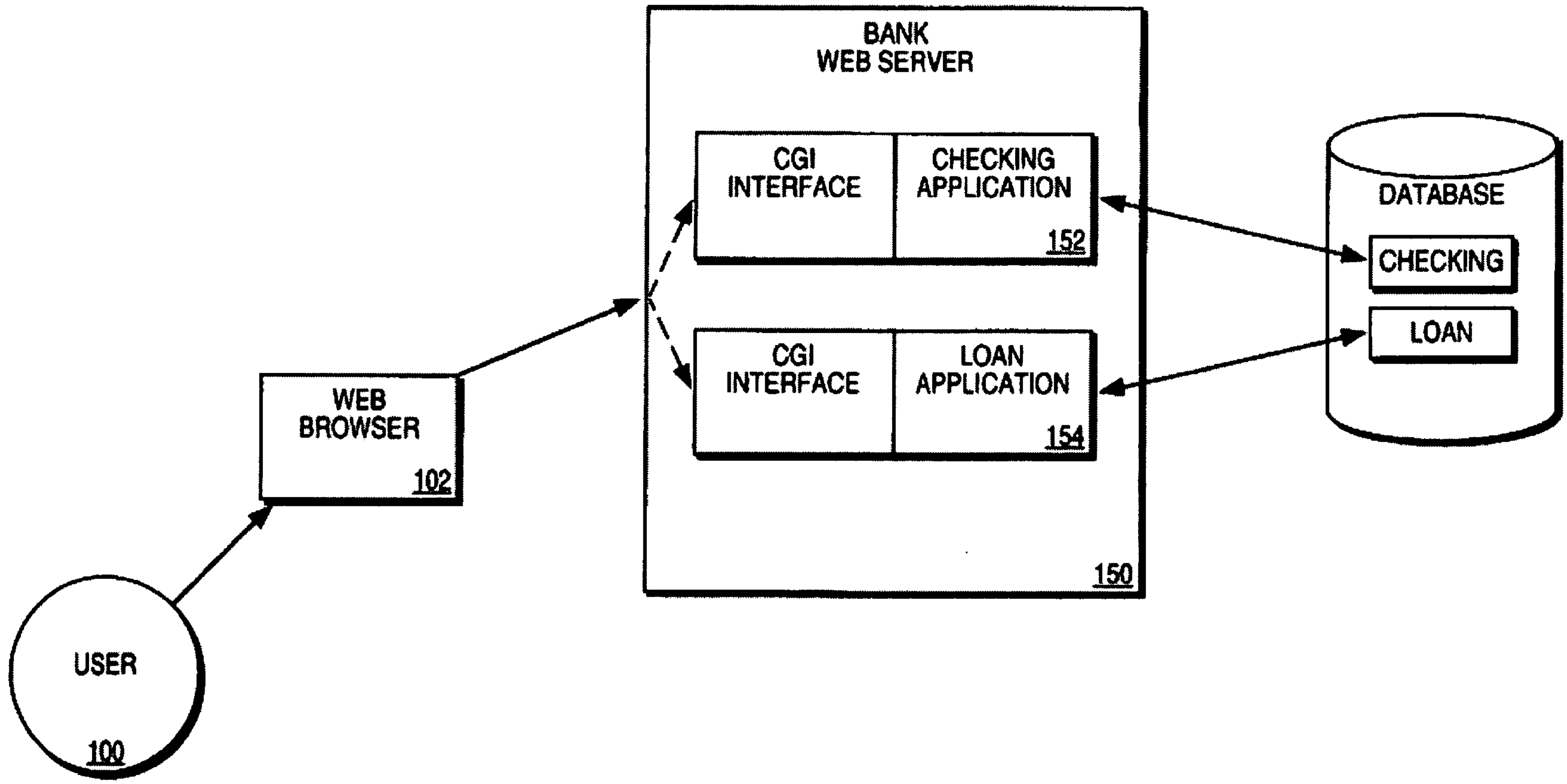


FIG. 1B (PRIOR ART)

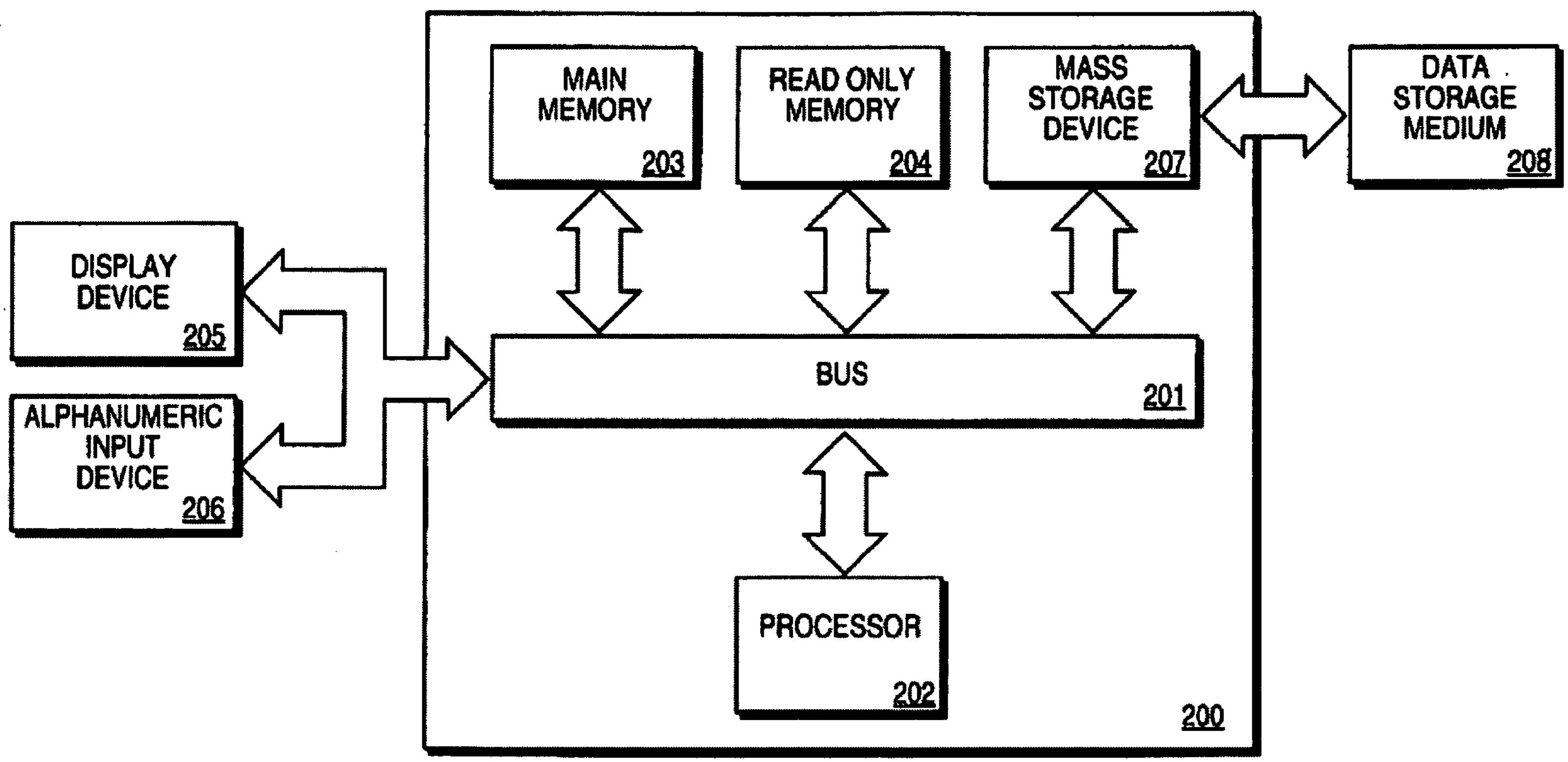

MAIN MEMORY 203
READ ONLY MEMORY 204
MASS STORAGE DEVICE 207
DATA STORAGE MEDIUM 208
DISPLAY DEVICE 205
ALPHANUMERIC INPUT DEVICE 206
BUS 201
PROCESSOR 202
200

FIG. 2

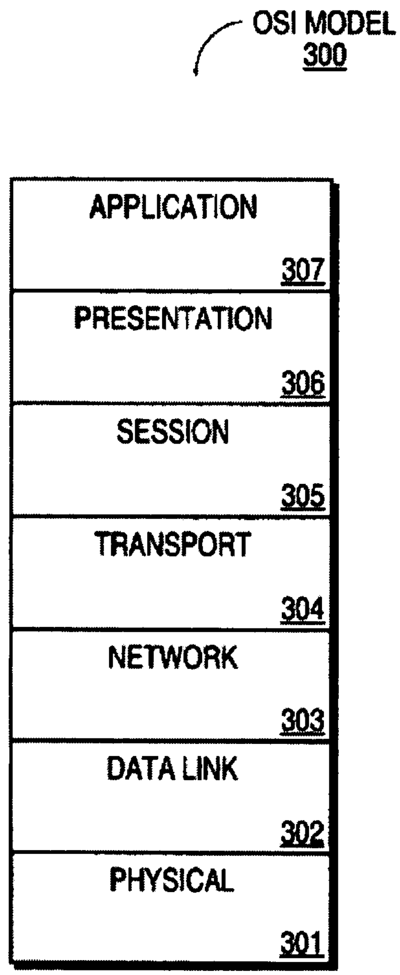

OSI MODEL
300
APPLICATION
307
PRESENTATION
306
SESSION
305
TRANSPORT
304
NETWORK
303
DATA LINK
302
PHYSICAL
301

FIG. 3

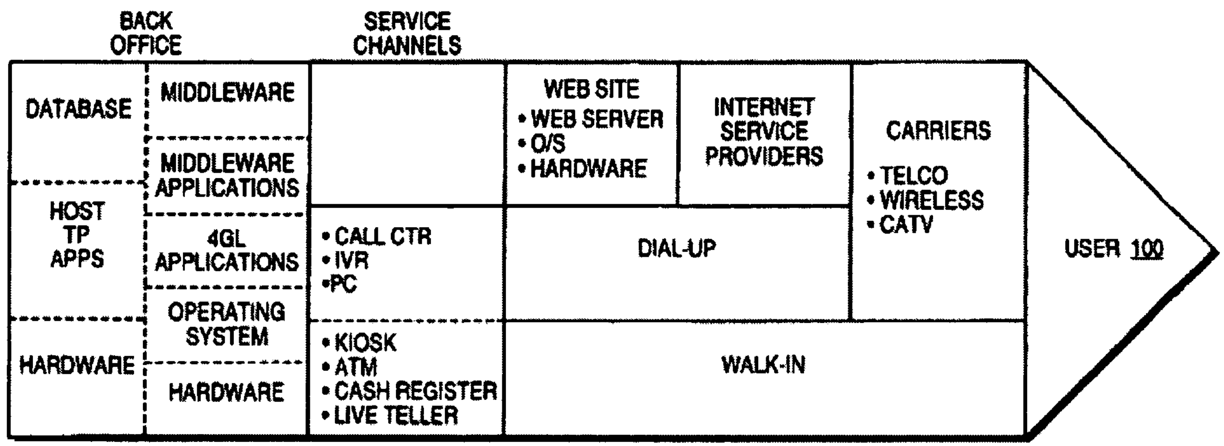

BACK OFFICE
SERVICE CHANNELS
DATABASE
MIDDLEWARE
MIDDLEWARE APPLICATIONS
HOST TP APPS
4GL APPLICATIONS
OPERATING SYSTEM
HARDWARE
HARDWARE
WEB SITE
• WEB SERVER
• O/S
• HARDWARE
INTERNET SERVICE PROVIDERS
CARRIERS
• TELCO
• WIRELESS
• CATV
• CALL CTR
• IVR
•PC
DIAL-UP
• KIOSK
• ATM
• CASH REGISTER
• LIVE TELLER
WALK-IN
USER 100

FIG. 4A

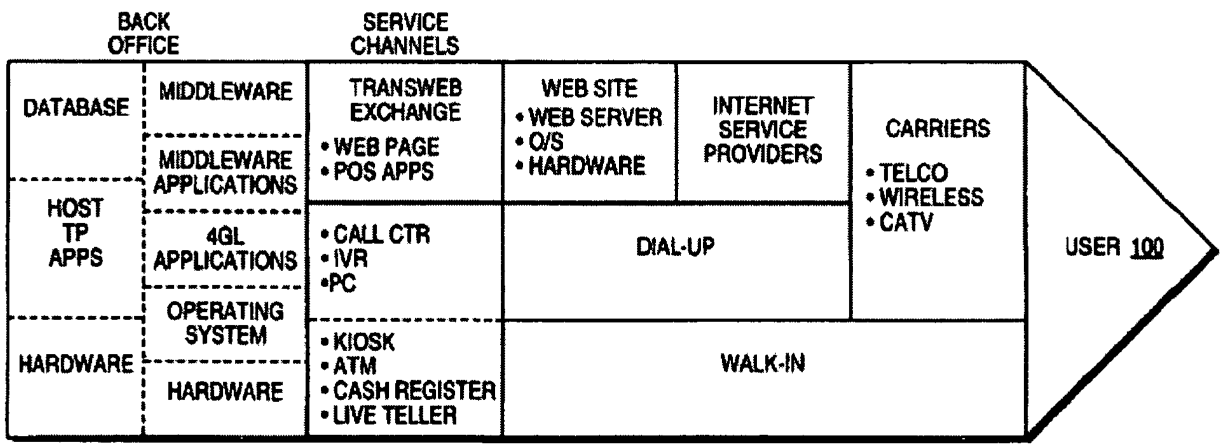

BACK OFFICE
SERVICE CHANNELS
DATABASE
MIDDLEWARE
MIDDLEWARE APPLICATIONS
HOST TP APPS
4GL APPLICATIONS
OPERATING SYSTEM
HARDWARE
HARDWARE
TRANSWEB EXCHANGE
• WEB PAGE
• POS APPS
WEB SITE
• WEB SERVER
• O/S
• HARDWARE
INTERNET SERVICE PROVIDERS
CARRIERS
• TELCO
• WIRELESS
• CATV
• CALL CTR
• IVR
•PC
DIAL-UP
• KIOSK
• ATM
• CASH REGISTER
• LIVE TELLER
WALK-IN
USER 100

FIG. 4B

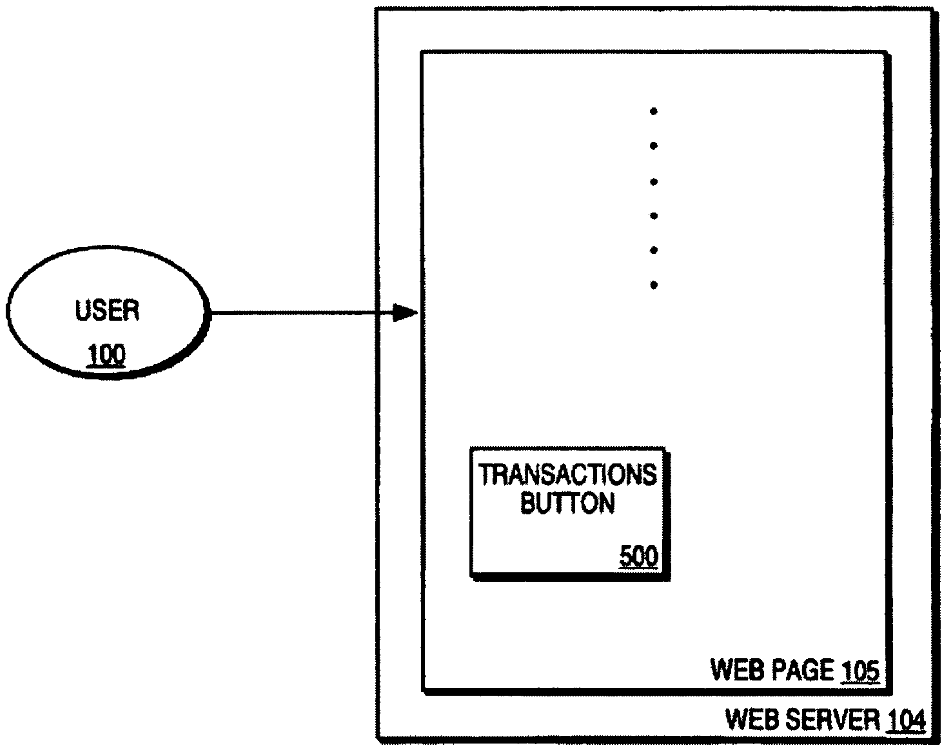

USER
100
TRANSACTIONS BUTTON
500
WEB PAGE 105
WEB SERVER 104

FIG. 5A

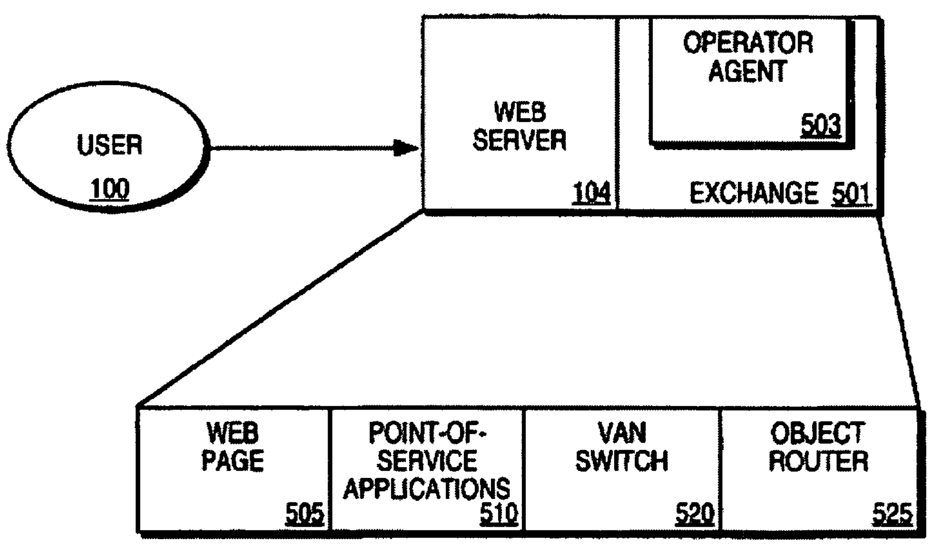

USER
100
WEB SERVER
104
OPERATOR AGENT
503
EXCHANGE 501
WEB PAGE
505
POINT-OF-SERVICE APPLICATIONS
510
VAN SWITCH
520
OBJECT ROUTER
525

FIG. 5B

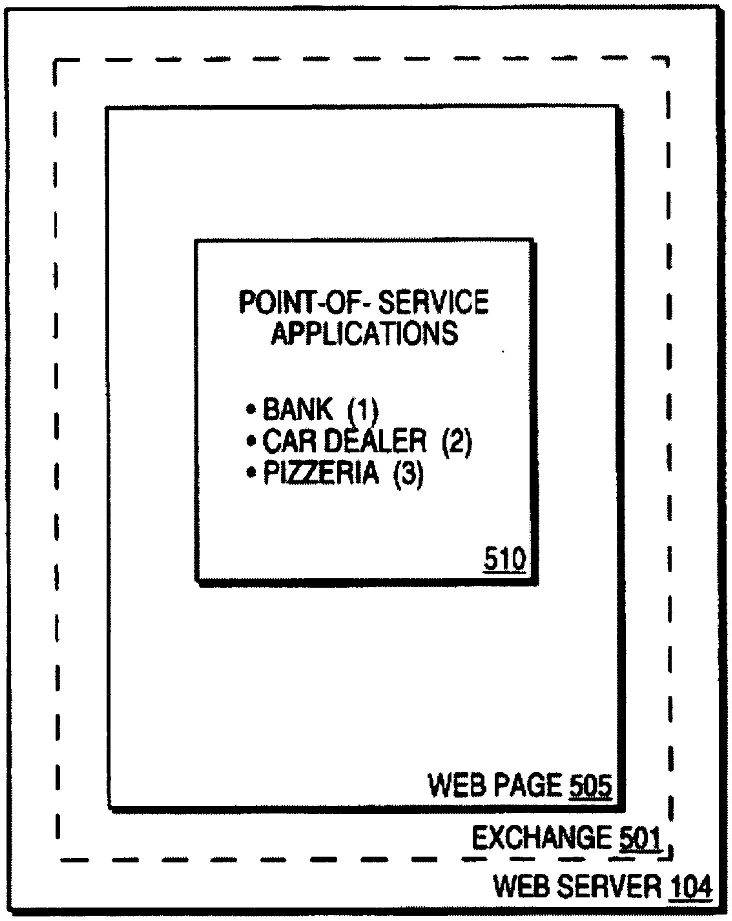

POINT-OF- SERVICE
APPLICATIONS
• BANK (1)
• CAR DEALER (2)
• PIZZERIA (3)
510
WEB PAGE 505
EXCHANGE 501
WEB SERVER 104

FIG. 5C

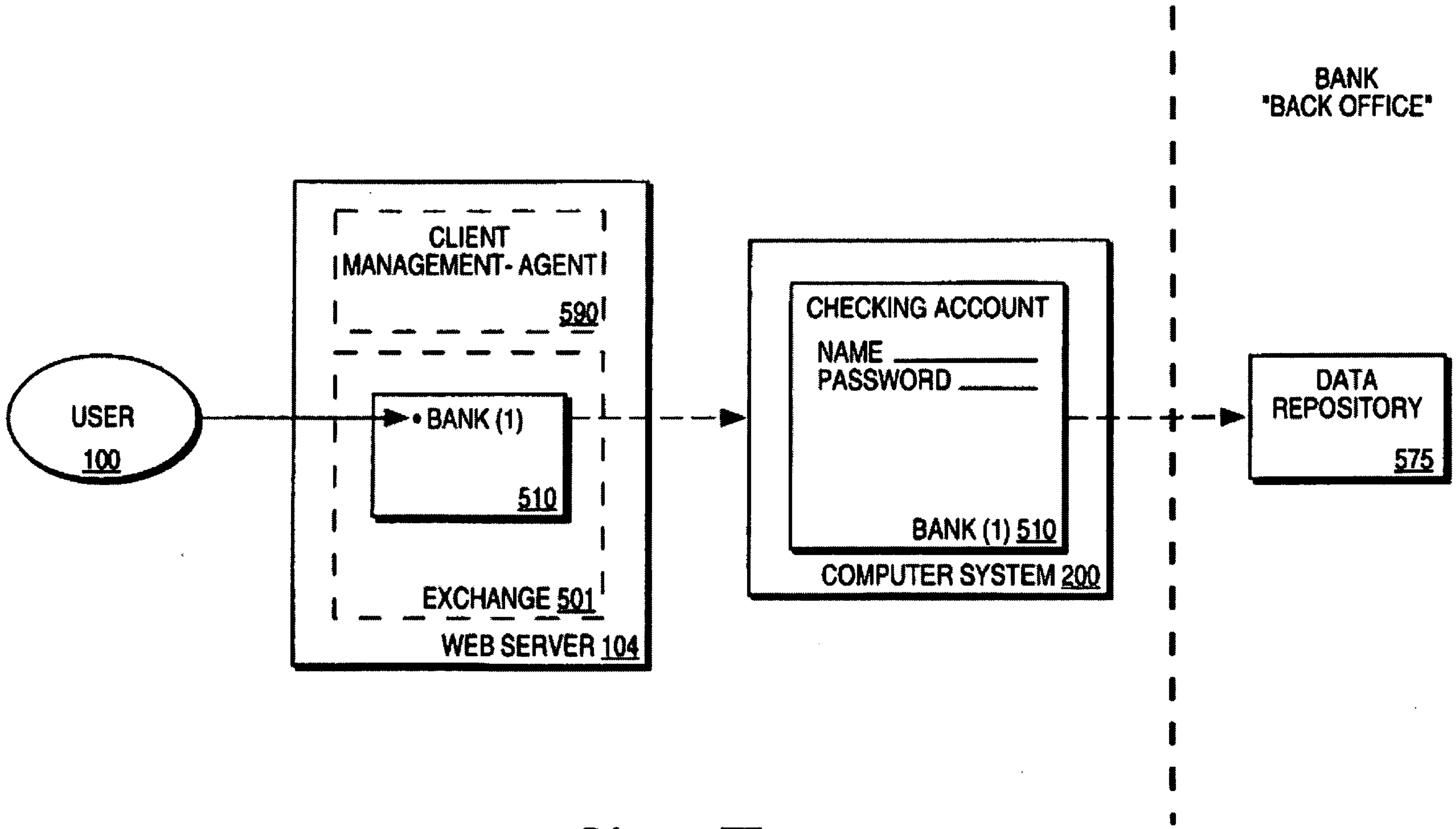

USER
100
CLIENT
MANAGEMENT- AGENT
590
BANK (1)
510
EXCHANGE 501
WEB SERVER 104
CHECKING ACCOUNT
NAME
PASSWORD
BANK (1) 510
COMPUTER SYSTEM 200
BANK
"BACK OFFICE"
DATA
REPOSITORY
575

FIG. 5D

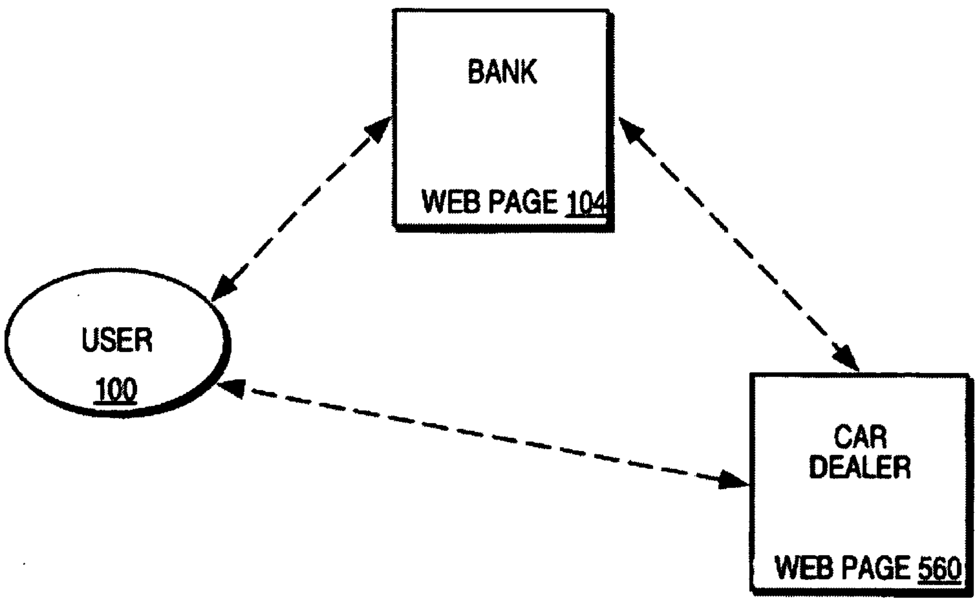


FIG. 5E

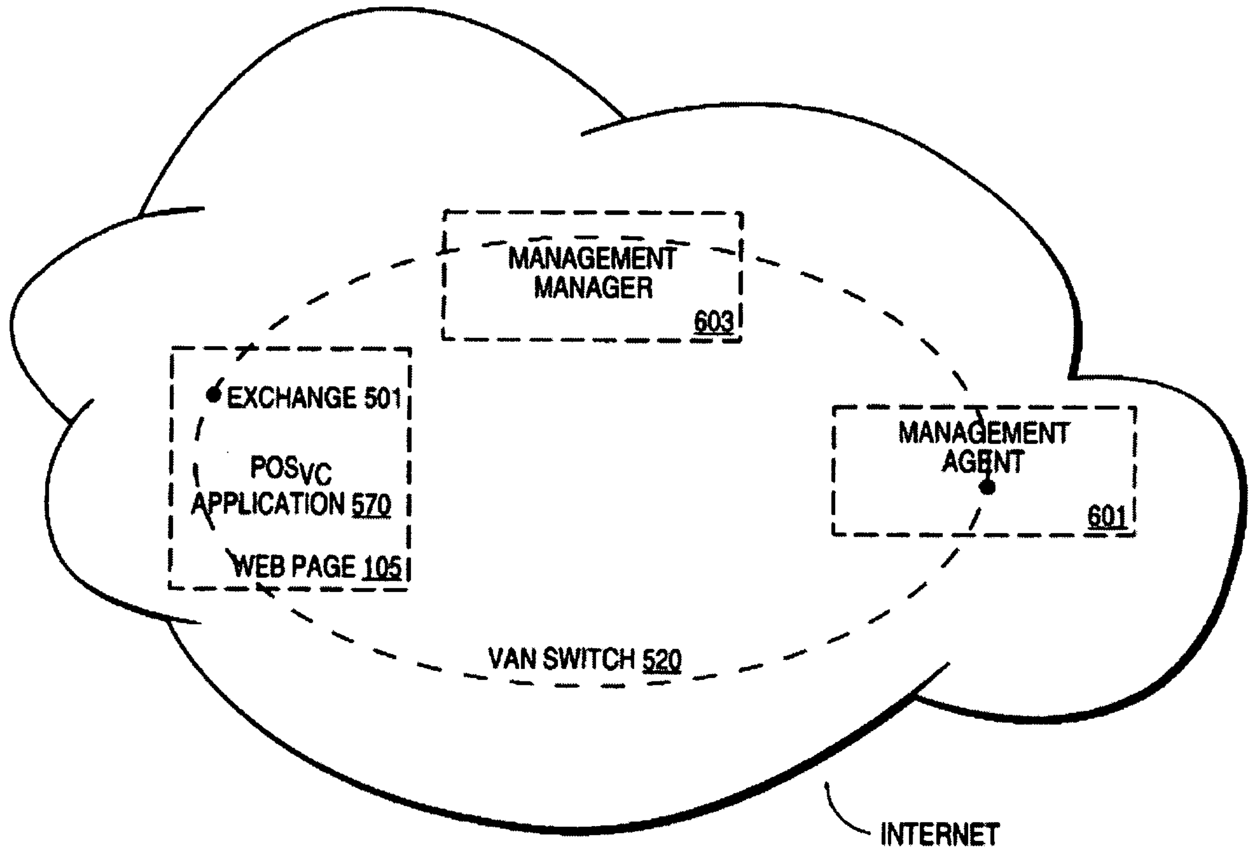

MANAGEMENT
MANAGER
603
EXCHANGE 501
POSVC
APPLICATION 570
WEB PAGE 105
MANAGEMENT
AGENT
601
VAN SWITCH 520
INTERNET

FIG. 6A

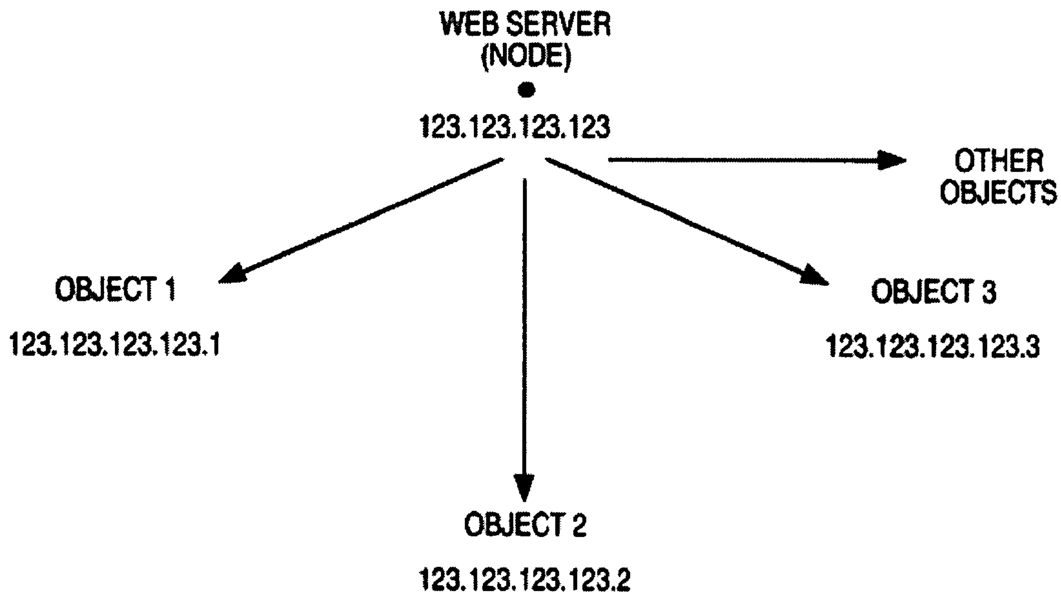

WEB SERVER
(NODE)
123.123.123.123
OTHER
OBJECTS
OBJECT 1
123.123.123.123.1
OBJECT 3
123.123.123.123.3
OBJECT 2
123.123.123.123.2

FIG. 6B

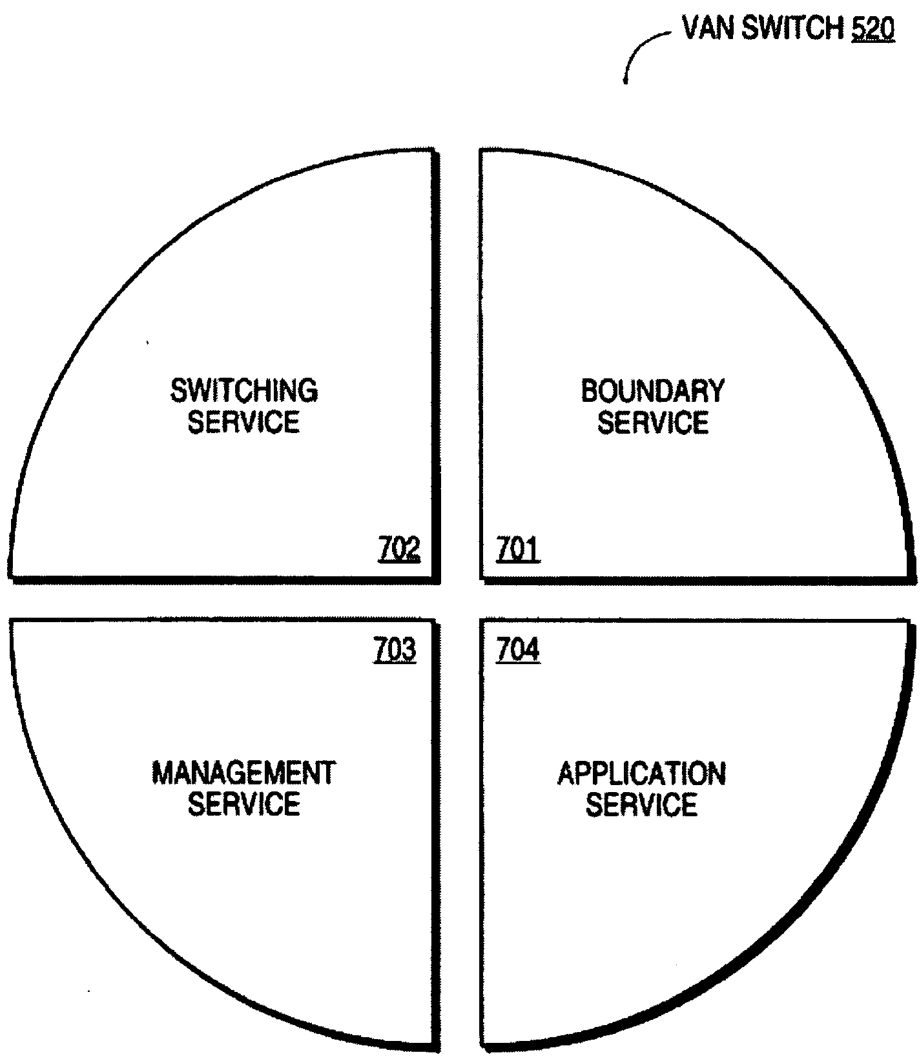

VAN SWITCH 520
SWITCHING SERVICE
702
BOUNDARY SERVICE
701
703
MANAGEMENT SERVICE
704
APPLICATION SERVICE

FIG. 7

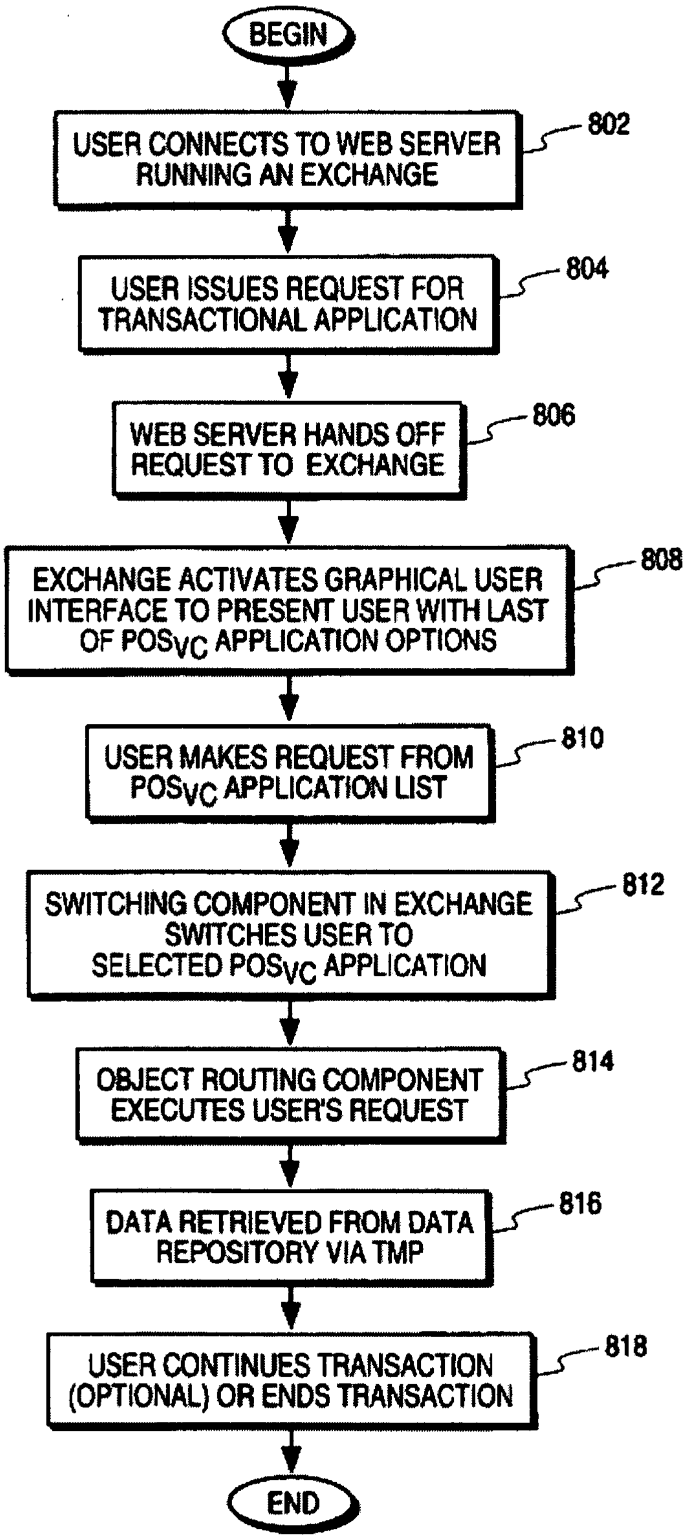

BEGIN
USER CONNECTS TO WEB SERVER RUNNING AN EXCHANGE
802
USER ISSUES REQUEST FOR TRANSACTIONAL APPLICATION
804
WEB SERVER HANDS OFF REQUEST TO EXCHANGE
806
EXCHANGE ACTIVATES GRAPHICAL USER INTERFACE TO PRESENT USER WITH LAST OF POSVC APPLICATION OPTIONS
808
USER MAKES REQUEST FROM POSVC APPLICATION LIST
810
SWITCHING COMPONENT IN EXCHANGE SWITCHES USER TO SELECTED POSVC APPLICATION
812
OBJECT ROUTING COMPONENT EXECUTES USER'S REQUEST
814
DATA RETRIEVED FROM DATA REPOSITORY VIA TMP
816
USER CONTINUES TRANSACTION (OPTIONAL) OR ENDS TRANSACTION
818
END

FIG. 8

10-30-07

PTO/SB/06 (12-04)
Approved for use through 7/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number: 11,980,185

APPLICATION AS FILED – PART I

FOR	NUMBER FILED (Column 1)	NUMBER EXTRA (Column 2)	SMALL ENTITY RATE ($)	SMALL ENTITY FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY FEE ($)
BASIC FEE (37 CFR 1.16(a), (b), or (c))				155			310
SEARCH FEE (37 CFR 1.16(k), (i), or (m))				255			510
EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))				105			210
TOTAL CLAIMS (37 CFR 1.16(i))	110	90	X$ 25	2250	OR	X$50	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	13 minus 3 =	* 10	X$105	1050		X$210	
APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR						
MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))			185			360	
			TOTAL	3815		TOTAL	

* If the difference in column 1 is less than zero, enter "0" in column 2.

APPLICATION AS AMENDED – PART II

AMENDMENT A

	CLAIMS REMAINING AFTER AMENDMENT (Column 1)		HIGHEST NUMBER PREVIOUSLY PAID FOR (Column 2)	PRESENT EXTRA (Column 3)	SMALL ENTITY RATE ($)	SMALL ENTITY ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	*	Minus	**	=	X =		OR	X =	
Independent (37 CFR 1.16(h))	*	Minus	***	=	X =		OR	X =	
Application Size Fee (37 CFR 1.16(s))									
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))					185		OR	360	
					TOTAL ADD'T FEE		OR	TOTAL ADD'T FEE	

OR

AMENDMENT B

	CLAIMS REMAINING AFTER AMENDMENT (Column 1)		HIGHEST NUMBER PREVIOUSLY PAID FOR (Column 2)	PRESENT EXTRA (Column 3)	RATE ($)	ADDITIONAL FEE ($)	OR	RATE ($)	ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	*	Minus	**	=	X =		OR	X =	
Independent (37 CFR 1.16(h))	*	Minus	***	=	X =		OR	X =	
Application Size Fee (37 CFR 1.16(s))									
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))					N/A		OR	N/A	
					TOTAL ADD'T FEE		OR	TOTAL ADD'T FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.