The Internet network management architecture is thus modular by design, with a protocol-independent data definition language and MIB, and an MIB-independent protocol. Interestingly, this modular architecture was first put in place to ease the transition from an SNMP-based network management to a network management framework being developed by ISO, the competing network management architecture when SNMP was first conceived—a transition that never occurred. Over time, however, SNMP's design modularity has allowed it to evolve through three major revisions, with each of the four major parts of SNMP discussed above evolving independently. Clearly, the right decision about modularity was made, even if for the wrong reason!

In the following subsections, we cover the four major components of the Internet-Standard Management Framework in more detail.

9.3.1 Structure of Management Information: SMI

The **Structure of Management Information, SMI** (a rather oddly named component of the network management framework whose name gives no hint of its functionality), is the language used to define the management information residing in a managed-network entity. Such a definition language is needed to ensure that the syntax and semantics of the network management data are well defined and unambiguous. Note that the SMI does not define a specific instance of the data in a managed-network entity, but rather the language in which such information is specified. The documents describing the SMI for SNMPv3 (which rather confusingly, is called SMIv2) are [RFC 2578; RFC 2579; RFC 2580]. Let's examine the SMI in a bottom-up manner, starting with the base data types in the SMI. We'll then look at how managed objects are described in SMI, then how related managed objects are grouped into modules.

SMI Base Data Types

RFC 2578 specifies the basic data types in the SMI MIB module-definition language. Although the SMI is based on the ASN.1 (Abstract Syntax Notation One) [ISO X.680 2002] object-definition language (see Section 9.4), enough SMI-specific data types have been added that SMI should be considered a data definition language in its own right. The 11 basic data types defined in RFC 2578 are shown in Table 9.1. In addition to these scalar objects, it is also possible to impose a tabular structure on an ordered collection of MIB objects using the SEQUENCE OF construct; see RFC 2578 for details. Most of the data types in Table 9.1 will be familiar (or self-explanatory) to most readers. The one data type we will discuss in more detail shortly is the OBJECT IDENTIFIER data type, which is used to name an object.

SMI Higher-Level Constructs

In addition to the basic data types, the SMI data definition language also provides higher-level language constructs.

Data Type	Description
INTEGER	32-bit integer, as defined in ASN.1, with a value between -2^{31} and $2^{31} - 1$ inclusive, or a value from a list of possible named constant values.
Integer32	32-bit integer with a value between -2^{31} and $2^{31} - 1$ inclusive.
Unsigned32	Unsigned 32-bit integer in the range 0 to $2^{32} - 1$ inclusive.
OCTET STRING	ASN.1-format byte string representing arbitrary binary or textual data, up to 65,535 bytes long.
OBJECT IDENTIFIER	ASN.1-format administratively assigned (structured name); see Section 9.3.2.
IPaddress	32-bit Internet address, in network-byte order.
Counter32	32-bit counter that increases from 0 to $2^{32} - 1$ and then wraps around to 0.
Counter64	64-bit counter.
Gauge32	32-bit integer that will not count above $2^{32} - 1$ nor decrease beyond 0 when increased or decreased.
TimeTicks	Time, measured in 1/100ths of a second since some event.
Opaque	Uninterpreted ASN.1 string, needed for backward compatibility.

Table 9.1 ♦ Basic data types of the SMI

The OBJECT-TYPE construct is used to specify the data type, status, and semantics of a managed object. Collectively, these managed objects contain the management data that lies at the heart of network management. There are more than 10,000 defined objects in various Internet RFCs [RFC 3410]. The OBJECT-TYPE construct has four clauses. The SYNTAX clause of an OBJECT-TYPE definition specifies the basic data type associated with the object. The MAX-ACCESS clause specifies whether the managed object can be read, be written, be created, or have its value included in a notification. The STATUS clause indicates whether the object definition is current and valid, obsolete (in which case it should not be implemented, as its definition is included for historical purposes only), or deprecated (obsolete, but implementable for interoperability with older implementations). The DESCRIPTION clause contains a human-readable textual definition of the object; this "documents" the purpose of the managed object and should provide all the semantic information needed to implement the managed object.

As an example of the OBJECT-TYPE construct, consider the `ipSystemStatsInDelivers` object-type definition from [RFC 4293]. This object defines a 32-bit counter that keeps track of the number of IP datagrams that were received at the managed device and were successfully delivered to an upper-layer protocol.

The final line of this definition is concerned with the name of this object, a topic we'll consider in the following subsection.

```
ipSystemStatsInDelivers OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS read-only
    STATUS      current
    DESCRIPTION
          "The total number of datagrams successfully
          delivered to IPuser-protocols (including ICMP).

          When tracking interface statistics, the counter
          of the interface to which these datagrams were
          addressed is incremented. This interface might
          not be the same as the input interface for
          some of the datagrams.

          Discontinuities in the value of this counter can
          occur at re-initialization of the management
          system, and at other times as indicated by the
          value of ipSystemStatsDiscontinuityTime."
    ::= { ipSystemStatsEntry 18 }
```

The MODULE-IDENTITY construct allows related objects to be grouped together within a "module." For example, [RFC 4293] specifies the MIB module that defines managed objects (including `ipSystemStatsInDelivers`) for managing implementations of the Internet Protocol (IP) and its associated Internet Control Message Protocol (ICMP). [RFC 4022] specifies the MIB module for TCP, and [RFC 4113] specifies the MIB module for UDP. [RFC 4502] defines the MIB module for RMON remote monitoring. In addition to containing the OBJECT-TYPE definitions of the managed objects within the module, the MODULE-IDENTITY construct contains clauses to document contact information of the author of the module, the date of the last update, a revision history, and a textual description of the module. As an example, consider the module definition for management of the IP protocol:

```
ipMIB MODULE-IDENTITY
    LAST-UPDATED "200602020000Z"
    ORGANIZATION "IETF IPv6 MIB Revision Team"
    CONTACT-INFO
          "Editor:
          Shawn A. Routhier
          Interworking Labs
          108 Whispering Pines Dr. Suite 235
```

```
        Scotts Valley, CA 95066
        USA
        EMail: <sar@iwl.com>"
    DESCRIPTION
        "The MIB module for managing IP and ICMP
        implementations, but excluding their
        management of IP routes.

        Copyright (C) The Internet Society (2006).
        This version of this MIB module is part of
        RFC 4293; see the RFC itself for full legal
        notices."

    REVISION      "200602020000Z"
    DESCRIPTION
        "The IP version neutral revision with added
        IPv6 objects for ND, default routers, and
        router advertisements. As well as being the
        successor to RFC 2011, this MIB is also the
        successor to RFCs 2465 and 2466. Published
        as RFC 4293."

    REVISION      "199411010000Z"
    DESCRIPTION
        "A separate MIB module (IP-MIB) for IP and
        ICMP management objects. Published as RFC
        2011."

    REVISION      "199103310000Z"
    DESCRIPTION
        "The initial revision of this MIB module was
        part of MIB-II, which was published as RFC
        1213."
    ::= { mib-2 48}
```

The NOTIFICATION-TYPE construct is used to specify information regarding SNMPv2-Trap and InformationRequest messages generated by an agent, or a managing entity; see Section 9.3.3. This information includes a textual DESCRIPTION of when such messages are to be sent, as well as a list of values to be included in the message generated; see [RFC 2578] for details. The MODULE-COMPLIANCE construct defines the set of managed objects within a module that an agent must implement. The AGENT-CAPABILITIES construct specifies the capabilities of agents with respect to object- and event-notification definitions.