

**Develop Unified SNMP, XML,
CLI, and Web-based
Management Easily With
MIBGuide™ Graphical IDE**

SNMP Research, Inc.

Knoxville, TN U.S.A.



**Copyright© 2005
SNMP Research, Inc.**

Topics

- **Multi-protocol accessible agent**
 - **EMANATE[®] vs EMANATE[®]/Lite agent**
 - **Protocols**
 - **SNMP (v1, v2c, v3, APO)**
 - **XML**
 - **CLI**
 - **HTTP/HTML**
 - **Other**
 - **Unified access**
- **MIBGuide**
 - **MIB Editor**
 - **MIB Compiler/source code generator**
 - **IDE**
 - **MIB Browser**
- **Q & A**



Multi-Protocol Accessible Agent

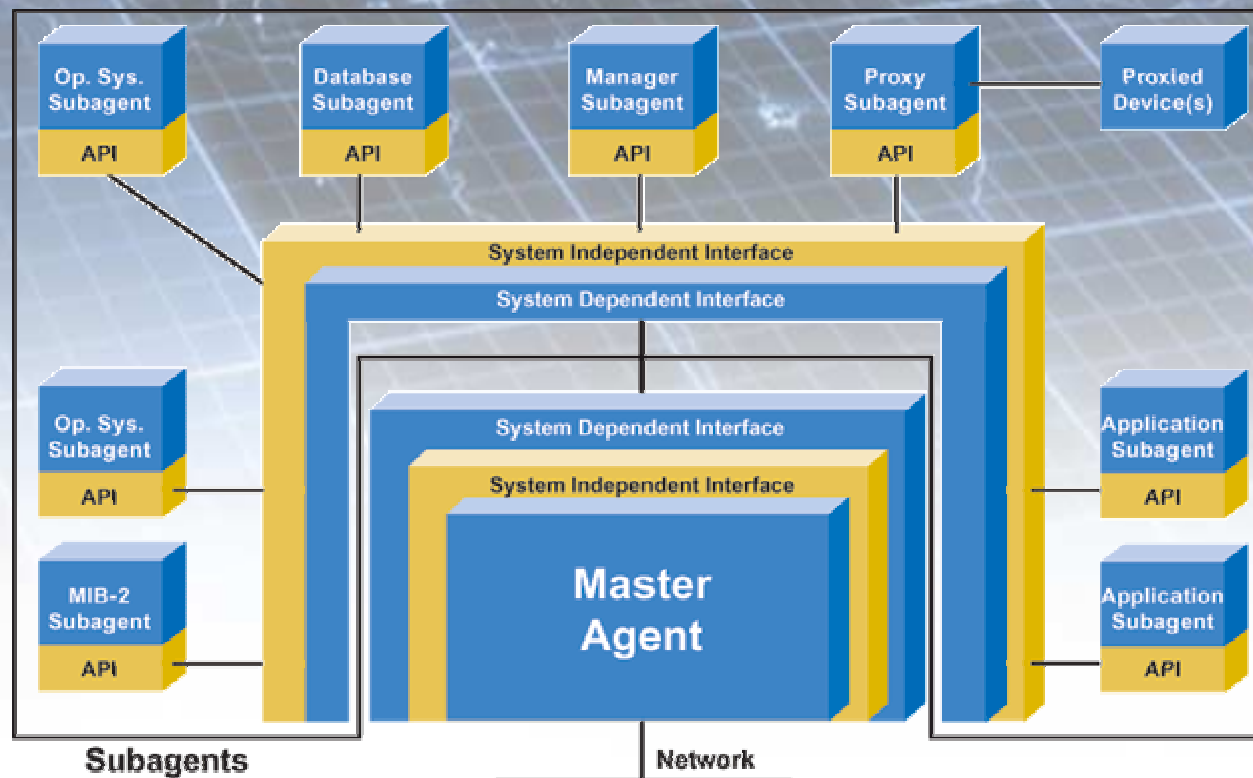
EMANATE vs EMANATE/Lite

EMANATE

- **Run-time Extensible Agent System**
- **Master Agent / Subagent paradigm**
- **Modularly constructed agent designed to address the problem of multiple agents on a single platform**
 - **Addresses the dynamic needs of open systems**
 - **Handles hot-swappable modules in embedded systems**

EMANATE Extensible Agent

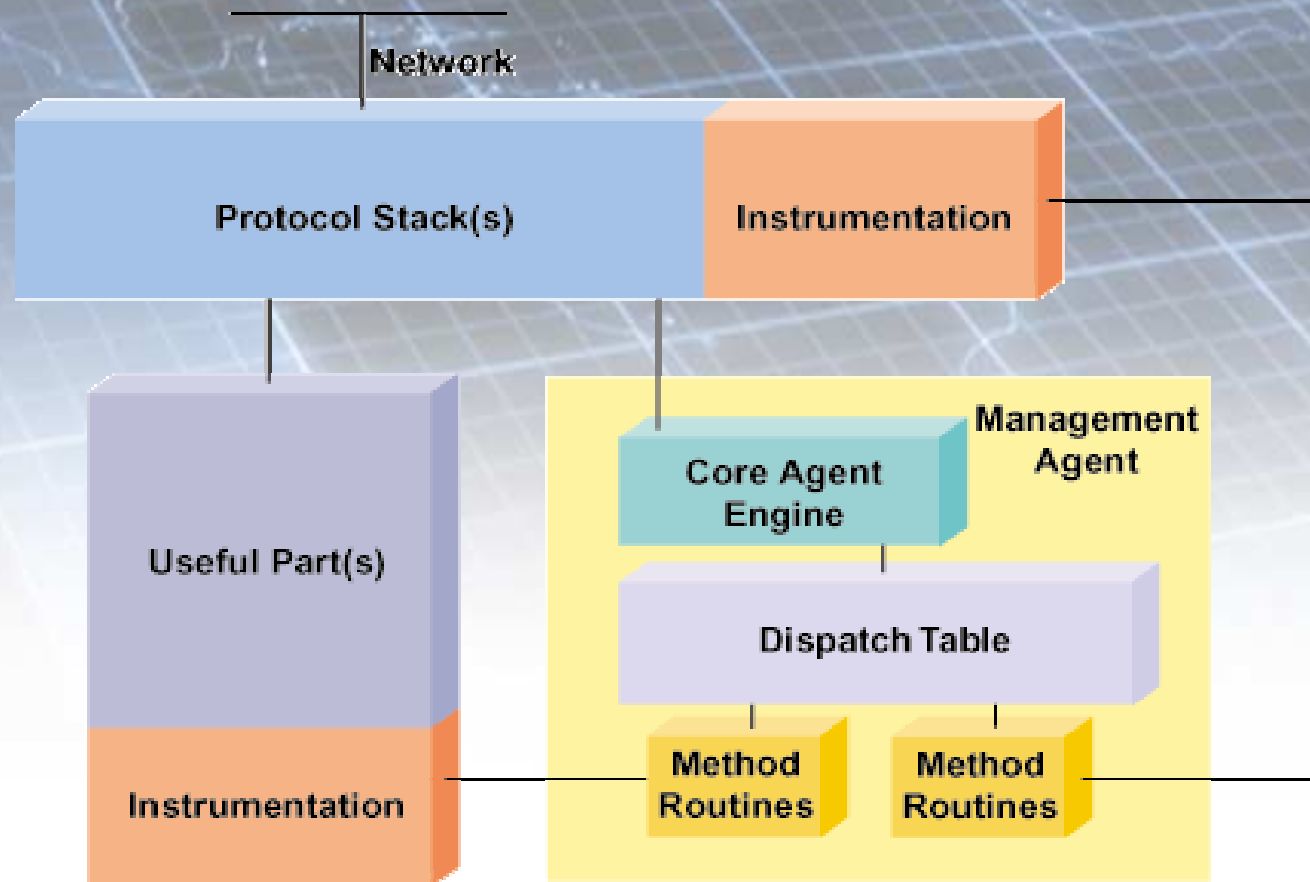
Master Agent /Subagent System



EMANATE/Lite

- **Compile-time Extensible Agent System**
- **Monolithic Agent paradigm**
 - **Low memory and CPU requirements**
 - **Single threaded Agent processes one PDU at a time**
 - **Available for many embedded systems**
 - **Source code based**

EMANATE/LITE





Multi-Protocol Accessible Agent

Protocols

SNMP

- **All agent products provide trilingual (SNMPv1, SNMPv2c, and SNMPv3) SNMP support**
- **Standards-compliant**
- **SNMPv3 Security**
 - **User-based security model**
 - **Authentication (MD5, SHA-1) and privacy (DES, Triple-DES, AES)**
- **SNMPv3 Administration**
 - **Fine-grained authorization and access control**
 - **Logical contexts**
 - **Remotely configurable via SNMP set operations**

SNMP (cont.) - Advanced Protocol Operations

- **APO Level 1: Compatible with SMIv2 MIBs**
 - **Allows for aggregate objects formerly inaccessible**
 - **Row Operations**
 - **Tabular Operations**
 - **OID Suppression**
 - **Improved write operations**
- **APO Level 2: Defines extended grammar**
 - **Allows for structs, unions, nesting, more expanded data types**
 - **Availability TBA**

XML

- **Agents support XML over SSL as alternate communication protocol**
- **Management through firewall support**
 - **TCP vs UDP**
 - **Dedicated port use**
 - **Connection-oriented**
- **Leverage off of existing XML-compliant applications**

CLI

- **Agents support Command-Line Interface**
- **“show” and “config” modes available for easy monitoring and configuration**
- **Enhanced printing options for scaling and arranging large amount of data**

HTTP/HTML

- **DR-Web technology allows authorized users to access the agent directly from a Web browser**
- **Automatically generates web pages for viewing/configuring MIB data. Custom pages are also supported**
- **Suited for customers seeking management solutions that are both standards-based and Web-browser-accessible**

Other Protocols via EPIC

- **Provides a solution for interfacing other protocols with EMANATE or EMANATE/Lite**
 - **TL1**
 - **CORBA**
 - **Java**
 - **Proprietary**



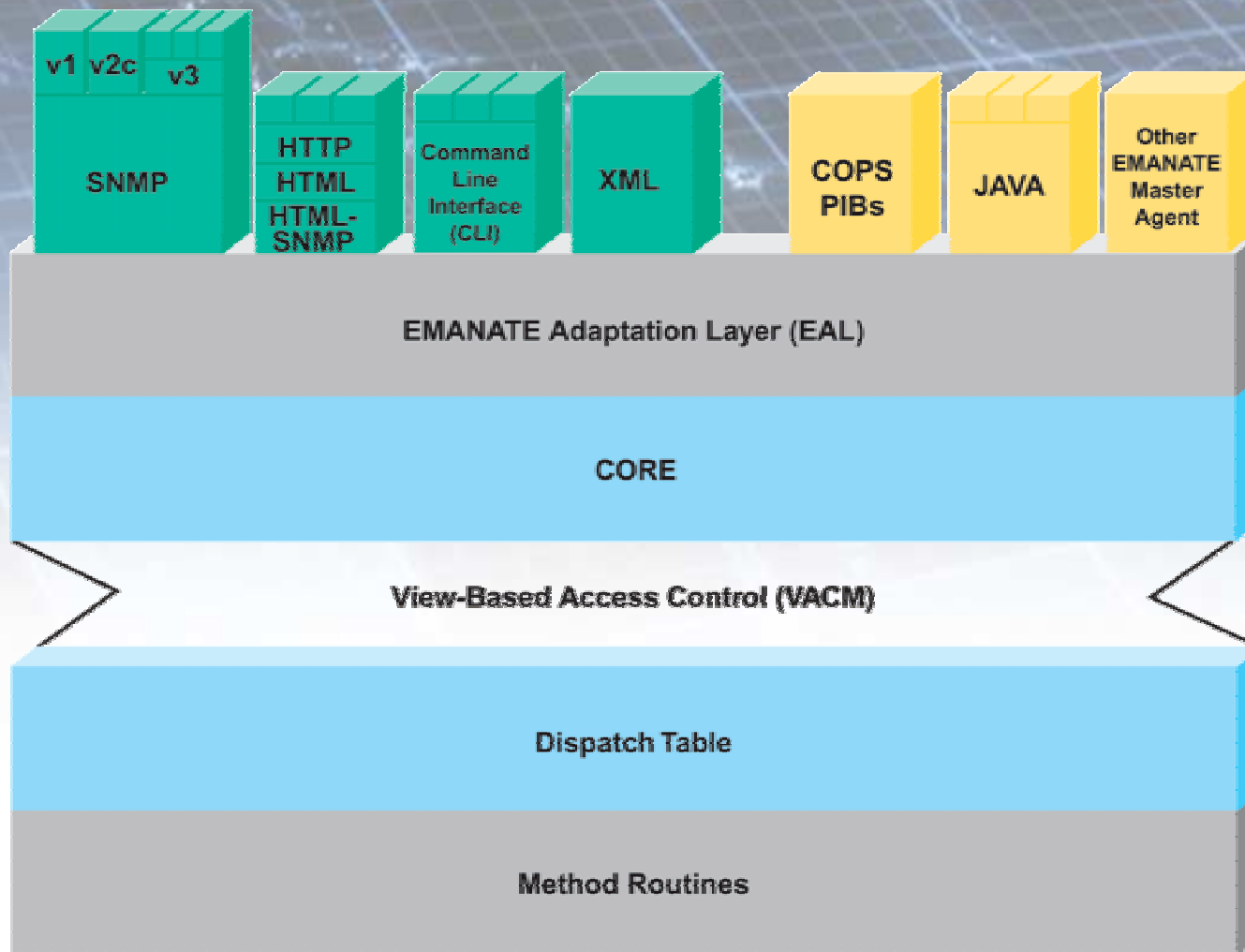
Multi-Protocol Accessible Agent

Unified Access

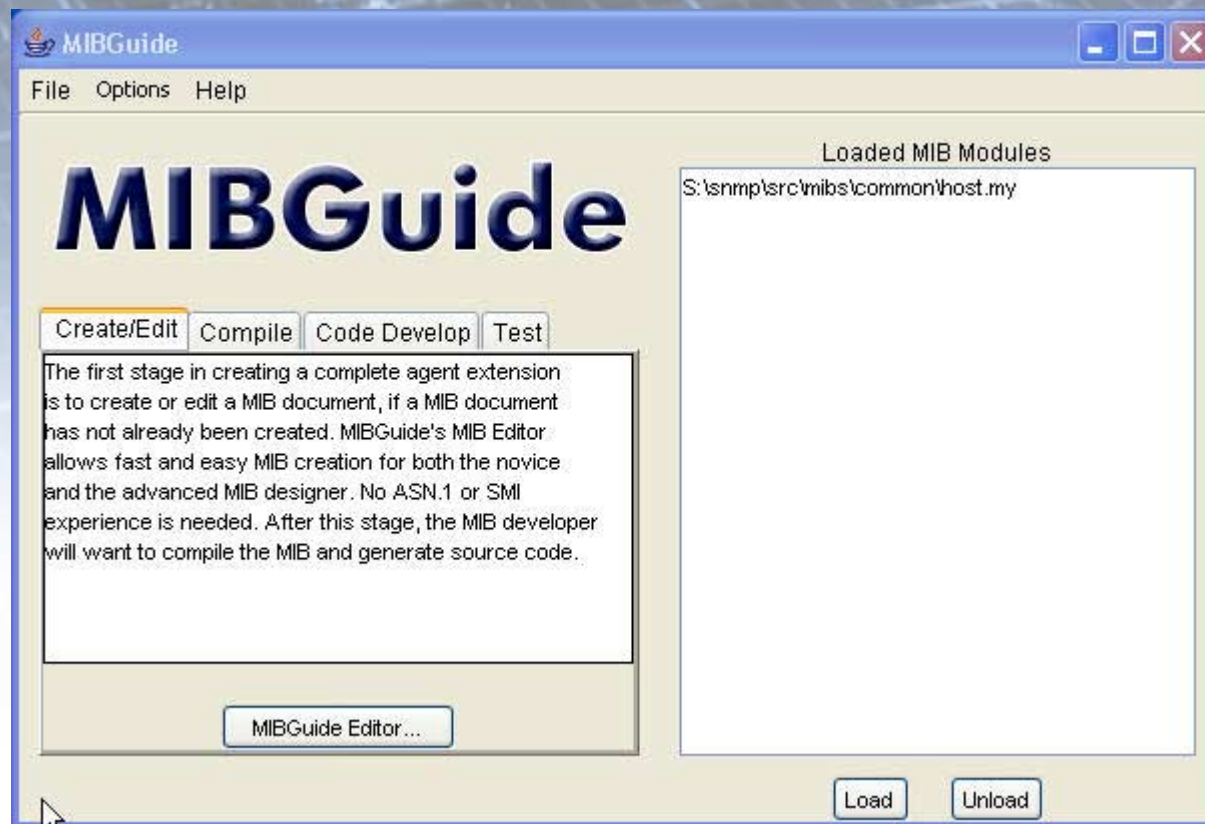
Unified Access

- **SNMP, XML, CLI, and Web all access the same data source**
 - **Write once, access multiple ways automatically**
 - **MIB-centric**
- **Access goes through a common security model**
 - **Allows administration of different security policies for each mode of access**

Multiple-Protocol Access Diagram



MIBGuide



Management Information Base Graphical User Integrated Development Environment

- **Included with both EMANATE and EMANATE/Lite**
- **GUI-based tool to ease the agent development process**
- **Productivity tool**
- **Reduces errors / increases compliance**
- **Reduces required skill**
- **Includes industry's leading MIB tools**

Agent Development Process

- **A 5 Step Process**
 - 1 **Define the MIB**
 - 2 **Compile the MIB (build skeleton code)**
 - 3 **Create the instrumentation**
 - 4 **Finish the access methods**
 - 5 **Test**

Step 1: Define MIB

MIBGuide MIB Editor

The screenshot displays the MIB Editor application window. The left pane shows a hierarchical tree of MIB modules, with the following structure expanded:

- MIBModule: HOST-RESOURCES-MIB
- IMPORTS
- ModuleIdentifier: hostResourcesMibModule
- ObjIdentifier: host
 - ObjIdentifier: hrSystem
 - Scalar: hrSystemUptime
 - Scalar: hrSystemDate
 - Scalar: hrSystemInitialLoadDevice
 - Scalar: hrSystemInitialLoadParameters
 - Scalar: hrSystemNumUsers
 - Scalar: hrSystemProcesses
 - Scalar: hrSystemMaxProcesses
 - ObjIdentifier: hrStorage
 - ObjIdentifier: hrStorageTypes
 - Scalar: hrMemorySize
 - Table: hrStorageTable
 - Row: hrStorageEntry
 - Column: hrStorageIndex
 - Column: hrStorageType
 - Column: hrStorageDescr (selected)
 - Column: hrStorageAllocator
 - Column: hrStorageSize
 - Column: hrStorageUsed
 - Column: hrStorageAllocator
 - ObjIdentifier: hrDevice
 - ObjIdentifier: hrDeviceTypes
 - Table: hrDeviceTable
 - Row: hrDeviceEntry
 - Column: hrDeviceIndex
 - Column: hrDeviceType

The right pane, titled "Column OBJECT-TYPE", shows the configuration for the selected column:

- Required Fields:**
 - Name: hrStorageDescr
 - Syntax: DisplayString (with Range... button)
 - MAX-ACCESS: read-only
 - STATUS: current
- Optional Fields:**
 - DESCRIPTION: A description of the type and instance of the described by this entry.
 - UNITS: (empty)
 - DEFVAL: (empty)
 - REFERENCE: (empty)
 - Comment: (empty)
 - OID Prefix: hrStorageEntry (with Subidentifier: 3)

Buttons for "Save" and "Reset" are located at the bottom of the right pane.

The bottom status bar displays the following text:

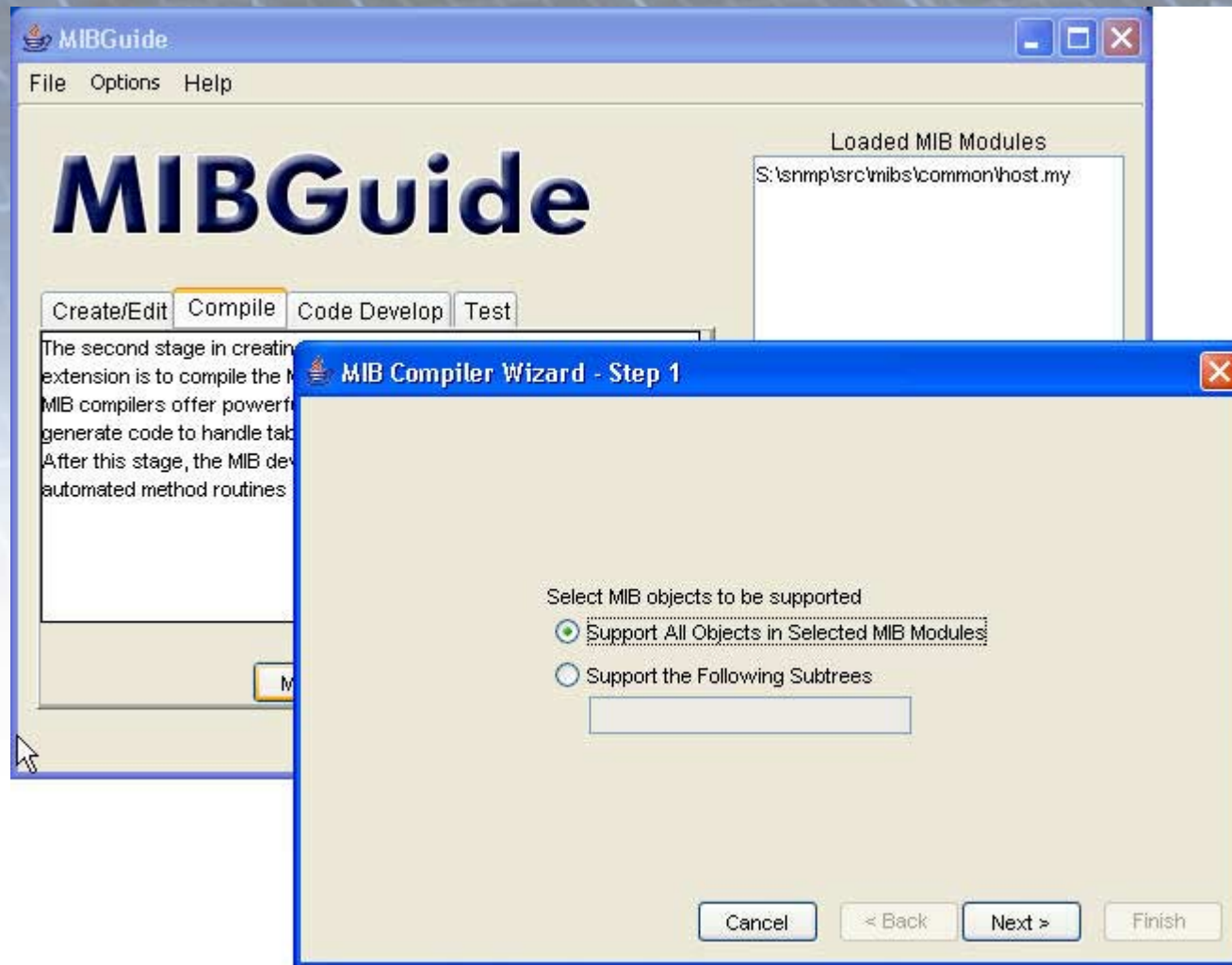
```
hrStorageDescr OBJECT-TYPE
SYNTAX DisplayString
MAX-ACCESS read-only
STATUS current
DESCRIPTION
    "A description of the type and instance of the storage
    described by this entry."
```

Step 1: Define MIB

MIBGuide MIB Editor

- **Allows both novice and advanced MIB designer an easy to use GUI for designing and importing MIB documents**
- **Checks for syntactic, semantic, and look-and-feel correctness**
- **Provides wizards and rapid prototyping tools**
- **Significantly reduces development time. In the past MIB development was the most difficult and time consuming task.**

Step 2: Compile MIB/Generate Code MIBGuide MIB Compiler



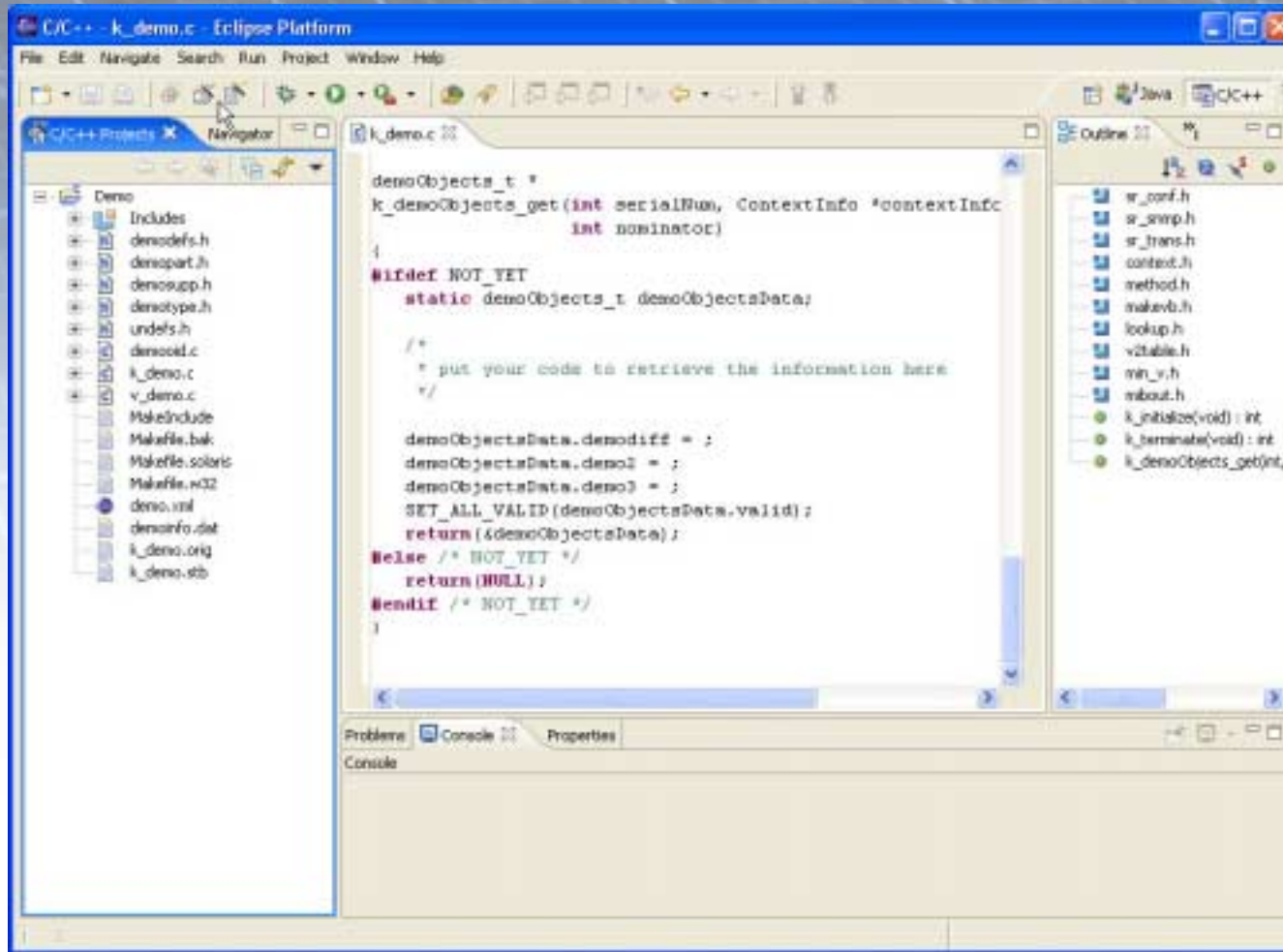
Step 2: Compile MIB/Generate Code MIBGuide MIB Compiler

- **Allows developers to select either C or JAVA API**
- **Can optionally automate code for other protocols (CLI, XML, Web, etc.)**
- **Provides choices for automatically generating code**
 - **Stable storage**
 - **Set routines**
 - **Robust table support**
 - **Many more**
- **Produces a working agent extension with no instances of any variable.**

Step 3: Create the Instrumentation

- **For example, add a gauge to indicate the number of valid entries in a database cache**
- **May be greatly simplified (or bypassed entirely) if augmenting a system where instrumentation already exists**
- **Outside the scope of MIBGuide**

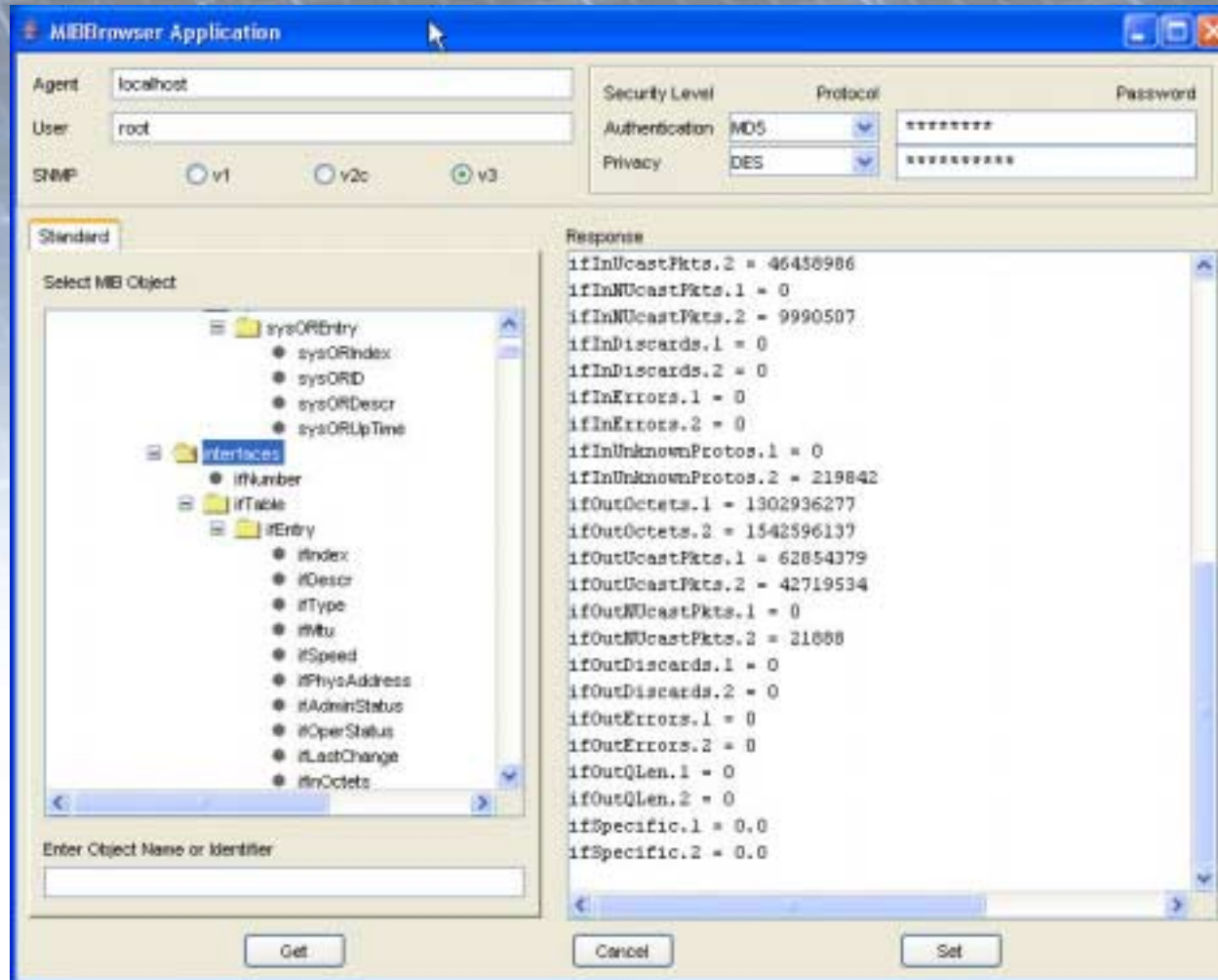
Step 4: Complete the Method Routines Integrated Development Environment



Step 4: Complete the Method Routines **Integrated Development Environment**

- **MIBGuide uses off the shelf IDE (IBM Eclipse, MS Visual Studio, etc.) for developer to modify code**
- **Complete the routines which support “get” operations as necessary**
 - **Remote monitoring of MIB objects will then be operational**
- **Complete the routines which support “set” operations**
 - **Remote configuration will then be operational**
- **Complete the routines which support “notify” operations**
 - **Notifications (traps) will then be supported**

Step 5: Test MIBGuide MIB Browser



Step 5: Test MIBGuide MIB Browser

- **Often the second most difficult step**
- **Testing the instrumentation is usually harder than testing the agent**
- **MIBGuide MIB Browser provides an easy way to “test as you go”**



Q & A

Booth 1118

www.snmp.com