## Table Of Contents

**AGENT TOOLKIT JAVA EDITION 6** ................................................................. 2  
**QUICK TOUR** ......................................................................................... 3  
  About Agent Toolkit Java Edition ......................................................... 4  
  Agent Architecture .............................................................................. 6  
  Agent Toolkit Features ....................................................................... 8  
  Agent Toolkit Java Edition Overview .................................................. 12  
    Working with Editor Tools ................................................................. 15  
    Working with Compiler tools ............................................................ 16  
    Working with Configuration Tools ................................................... 17  
    Working with Management Tools ..................................................... 18  
  Agent Toolkit Java Edition Architecture ............................................ 20  
    Multi-Protocol Agent Architecture .................................................. 20  
    SNMP Agent Architecture .............................................................. 31  
    TL1 Agent Architecture ................................................................. 33  
  AdventNet's Other Products ............................................................... 35  
  Contacting AdventNet ......................................................................... 37  
**INSTALLATION GUIDE** ....................................................................... 39  
  System Requirements .......................................................................... 40  
  Installation and Setup .......................................................................... 41  
  Starting Agent Toolkit .......................................................................... 42  
  Using License Manager ........................................................................ 43  
  Using Launcher ..................................................................................... 47  
  Using Update Manager ........................................................................ 49  
  Getting Started with Tools ................................................................... 54  
  Trouble Shooting at Startup ................................................................. 58  
  FAQ - Installation and Startup ............................................................. 60  
**RELEASE NOTES** ........................................................................... 64  
  Release Features .................................................................................. 65  
  Known Issues and Limitations ............................................................. 75  
  Bugs Fixed ............................................................................................ 77  
  Package Structure ................................................................................ 79  
  Agent Toolkit Jars Description ............................................................ 82  
  Third Party Software ............................................................................ 84  
**BUILDING SNMP AGENTS** ............................................................... 85  
**BUILDING MULTI-PROTOCOL AGENTS** ........................................... 86  
**BUILDING TL1 AGENTS** ................................................................. 87
The AdventNet Agent Toolkit Java Edition is a rapid prototyping and development tool used for building Java-based Standalone SNMP and TL1 agents. It also supports building agents with multi-protocol access to common management information through SNMP, RMI, HTTP, CORBA, and TL1. You just have to Instrument once to access through multi-protocols.

<table>
<thead>
<tr>
<th>Quick Tour:</th>
<th>It provides quick reference to AdventNet Agent Toolkit Java Edition which also includes its architecture and features. Quick tour also highlights details about contacting AdventNet for technical support and AdventNet's Other Products. Please go through Quick Tour section to make sure you are using the supported JDK platform and other products to develop your network management applications.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation Guide:</td>
<td>This provides information on how to install and start the product.</td>
</tr>
<tr>
<td>Release Notes:</td>
<td>This explains the functions and features of with the current release of the product. Information about the identified bugs and their fixes is also provided. Package Structure lets you know how the Agent Toolkit is bundled. Knowledge about the Third Party Software is given, too.</td>
</tr>
<tr>
<td>Building SNMP Agents:</td>
<td>This describes how to develop SNMP Agents. It also explains the SNMP Agent architecture, the various features supported, and its implementation by the Toolkit.</td>
</tr>
<tr>
<td>Building Multi-Protocol Agents:</td>
<td>This section provides you with a complete reference for building Multi-Protocol Agent, which includes its architecture, features, and its implementation.</td>
</tr>
<tr>
<td>Building TL1 Agents:</td>
<td>This guide helps the user to develop the TL1 agent using the tools provided. It also highlights its architecture, features, and implementation.</td>
</tr>
</tbody>
</table>

Java, JMX and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. or other countries
2.0 Quick Tour

About Agent Toolkit Java Edition: This section gives a complete reference of AdventNet Agent Toolkit. It introduces you to the features and concepts used in the product. The standards compatibility of the tool is also covered in this section.

Agent Architecture: This provides an overview of agent architecture that is supported by the Agent Toolkit.

Agent Features: It briefly explains the features supported by the different agents of Agent Toolkit.

Installation and Getting Started: This section helps you to know the basic requirements to run AdventNet Agent Toolkit Java Edition. Procedure to install and uninstall the toolkit is specified here. It also talks about the jars packaged with the toolkit.

Agent Toolkit Java Edition Overview: It assists you to get familiar with the development and testing tools bundled with the product for building the agents.

Agent Toolkit Java Edition Architecture: This section leads you to the diagrammatical representation of architecture of various agents that can be developed using Agent Toolkit. These architectures will give an high level picture of the agents.

AdventNet's Other Products: AdventNet provides a comprehensive range of network management products to meet all the network and system management needs of today's businesses. This page lists the various products and tools of AdventNet.

Contacting AdventNet: AdventNet is always eager to hear its customer's feedbacks and comments. This section gives the contact address and mail ID of Agent Toolkit support and sales teams.
About Agent Toolkit Java Edition

- About Agent Toolkit Java Edition
- Market Segments
- Key Features
- Product Architecture

About Agent Toolkit Java Edition

Most of the management systems are built from top to bottom either by SNMP, TL1 or HTTP. They leverage the specific facilities offered by the selected management protocol. For example, application server vendors are keen at SNMP for their management needs. Similarly TL1 is a man-machine management protocol deployed in broadband and access networks, including SONET networks. New technologies such as WDM and xDSL sometimes use the TL1 protocol for management of network equipment.

But these emerging OEM/Application/Enterprise vendors are worried about the rising cost of software development and the bearing on each operating system and platform to be supported. This has forced them to look for productivity enhancing technologies. And in the current fast paced environment, it is not practical for most application developers to support multiple platforms and operating systems in a timely manner. A cross platform, standard compliant management solution is required to allow rapid development of agents, for their applications, making them suitable for network management.

These needs are fulfilled by AdventNet Agent Toolkit Java Edition which provides the necessary infrastructure, with a comprehensive suite of Java based cross platform development tools, framework and APIs to be used across many operating systems, thus addressing the needs of fast paced environment for developing application/system management agents. It also offers investment proof and overall lower total cost of ownership including lower cost of development due to availability of skilled people and tools.

The AdventNet Agent Toolkit Java Edition is a rapid prototyping and development tool used for building Java based Standalone SNMP and TL1 agents. It also supports building agents with multi-protocol access to common management information through SNMP, RMI, HTTP, CORBA and TL1. You just have to instrument once to access through multi-protocols.

Market Segments

This product is well suited for the following market segments:

- Application management
- OEM vendors
- Enterprises
- System management
- Middleware ISV’s
- Service Providers
Key Features

The key features of AdventNet Agent Toolkit Java Edition which fulfills above market requirements are:

- **Open Standards**: Agent Toolkit Java Edition is built on standard technologies like JMX, Java Beans, XML, JFC, JDBC, HTTP, RMI, CORBA, SNMP, TL1, etc. Given the power and ease-of-use of these new Internet technologies, developers can now provide better solutions faster and at a lower cost of ownership including lower cost of development than before.

- **Cross Platform**
- Solaris, Windows NT/2000, and LINUX supported out-of-the-box.

- **Management Protocols**.

- **Agent Toolkit Java Edition** supports standalone SNMP and TL1 agents along with multi-protocol agent.

- **Development and test tools**: Agent Toolkit Java Edition provides comprehensive set of UI (editors, compilers and browsers) and command line tools for easy and faster development and testing of agents.

Product Architecture

The following diagram explains the architecture and tools available with the Agent Toolkit Java Edition for building cross platform Java agents:

![Fig.1-AdventNet Agent Toolkit Java Edition Product Overview](image-url)
Agent Architecture

- Agent Architecture - An Overview
- Single Agent Architecture
- Distributed Agent Architecture

Agent Architecture - An Overview

AdventNet Agent Toolkit Java Edition enables the user to build high performance, cross-platform agents for various type of application considering the key factor that different protocols are best suited for different needs. For example, SNMP is best suited for performance management and TL1 is best for fault management. That is performance monitoring applications can use SNMP agents and fault management applications can use TL1 agents.

This section explains the different agent architectures, single agent and distributed agents (master-subagent) architecture, supported by the AdventNet Agent Toolkit Java Edition.

Single Agent Architecture

Single agent architecture is preferable, when you are looking to build an agent to manage an:
- application running in a single JVM,
- application running in a single pc (system),
- embedded application.

The agent access to the application data (instrumentation), could be any one of the following:
- direct API call.
- proprietary protocol between the agent and the application.
- direct access to the database, text files used by the application.

Agent Toolkit provides all the above ways of instrumentation. The compilers has high degree of control and flexibility to generate stub files to achieve any of the above means of instrumentation.

Fig.1- Single Agent Architecture
Distributed Agent Architecture

Distributed agent (master - sub agent) architecture is preferable, when you are looking to build an agent to manage a distributed application where each of its components may be present in the same system or distributed across the network. In such distributed environment, each component may be treated as a separate agent (sub agent). However there should be a single gateway for managing the entire application, this would be the master agent.

![Distributed Agent Architecture](image)

Fig.2 Distributed Agent Architecture
Agent Toolkit Features

- Multi-Protocol Agent Features
- SNMP Agent Features
- TL1 Agent Features
- General Features

Multi-Protocol Agent Features

The following is a list of Multi-Protocol Agent features available with the AdventNet Agent Toolkit release 6.0.0:

- **Visual JMX Compiler** provides an integrated environment to generate, instrument, and compile the code.
- **JMX-SNMP Proxy feature** provides multi-protocol access to legacy SNMP agents, with just a code generation setting in JMX Compiler.
- **SNMP Adaptor** that provides complete SNMPv1, SNMPv2c, and SNMPv3 support.
- **Discovery Service** for automatic discovery of other agents in the network.
- **Cascading Service** that caters highly scalable Master agent to handle any number of subagents.
- **Authentication support** has now been provided for RMI Adaptor, CORBA Adaptor, and HTTP Connector in which RMI and CORBA are exposed as Service MBeans.
- **Relation Service** for logical association of MBeans.
- **Timer Service** for scheduling Notifications.
- **Monitor Service** to monitor MBean Attributes.
- Dynamic downloading of MBeans using MLet service.
- **Database Service MBean** to manage Database tables.
- **View Based Authorization** MBean for providing granular authorization for security.
- **Table Monitor** for monitoring table attributes of MBeans in addition to the existing support for monitoring String, Counter, and Gauge attributes.
- **HeartBeat support** for the Connector Clients.
- **Generic Adaptor - Client Framework** to build new protocol Adaptors and clients.
- **User storage model** that facilitates the user to have his own storage mechanism instead of agent storing the data.
- **FileToVector** or **XMLToVector files** that expose the application tabular data from the agent to the manager.
- Useful **Test tools** such as MBean Browser, MIB Browser and TL1 Browser make the testing easier, ensuring the complete agent development cycle.
- **Client API** for RMI, HTTP, and CORBA protocol connectors.
- **RMI Adaptor**, **HTTP Connector** and **HTML Adaptor** now support communication using SSL (Secure Sockets Layer).
- **Security support** in TL1 Adaptor.
- **Logging feature** has been provided to generate log messages at agent run time. Specific logging levels can also be specified.
SNMP Agent Features

- Complete SNMPv1, SNMPv2c, and SNMPv3 support.
- Highly scalable Master Agent to handle any number of Subagents.
- Intuitive visual MIB Editor for creating/editing SNMP MIB.
- High degree of control and flexibility in the generated source code.
- Support for Traps, Notifications, and Informs.
- Provides MIB Compiler to generate and compile the code to form an agent.
- Support for Access Control to authenticate read and read-write access to managers.
- USM and VACM support in SNMPv3. SNMPv3 also supports Co-existence and Notification Filtering Mechanism.
- Provides SNMPv3 administration tool for managing USM and VACM tables.
- Provides Rule Engine to define users' logic as rules to monitor the attributes of user's application/device and then execute actions.
- Support for IPv6.
- Agent designed to be ported in J2ME CDC environment.
- Support for running Multiple Agents in Single JVM.
- Transport Provider Framework Support.
- Supports Atomicity.
- Agents for Embedded JVM.
- Exposing application tabular data directly through the agent (no data duplication). This approach is named as "user storage model", where user can have their own storage mechanism instead of agent storing the data.
- Project workspace for Agents in MIB Compiler UI.

TL1 Agent Features

- TL1 Agent conforms to Bellcore GR 831, 815, 833 and TR 835 standards. Also provisioning messages from GR199 has been implemented.
- Security: Security feature in TL1 Agent with rich set of security options has been implemented as per the GR 815 and TR 835 standards.
- Asynchronous Message Processing: This enables the client to send many requests simultaneously without having to wait for the previous message to be processed.
- Delayed Activation Support enables an input message to be stored in a message pending buffer at the NE for final execution later.
- Built-in Commands support helps you control the TL1 Agent with much ease.
- Built-in Commands for Delayed Activation support helps you activate, deactivate, or retrieve Delay Activation input messages.
- Transport Provider Interface support helps you plug in any transport protocol.
- Self Monitoring support helps you monitor the Agent details such as the number of requests received, number of responses and autonomous messages sent, etc.
- Persistence Storage support that enables persistent storage for non-volatile information in the TL1 Agent.
- A TCS file has been defined for all the commands in GR 833.
- Multiple Port Support: TL1 Agent can be made to run on multiple ports simultaneously.
- Craft Interface Support: The input commands can be typed on the console from where they are sent to the TL1 Agent for processing.
- Error Code Customization: Using this feature, you can customize the error codes and their descriptions that are received by the manager as response.
General Features

- **Instrument Once Access with Any Protocol:** The Multi-Protocol framework allows to add protocol adaptors/connectors to the framework without disturbing the instrumentation done earlier. The Multi-protocol provides APIs to access the MBean (instrumentation) information which can be used by the user-defined protocol adaptor to be added to the framework. Also, this newly developed protocol adaptor can also be added to the agent on the fly using MLet (Management Applet) service, thus ensuring the dynamic behavior, a key factor for network management.

![Diagram of Instrument Once Access with Any Protocol](image)

**Fig.4 Instrument Once Access with Any Protocol**

- **Distributed Application Management:** The Agent Toolkit Java Edition provides a comprehensive solution for developing agents for distributed application. The master-subagent deployment architecture can be adopted for managing such distributed applications. Each component/service in the application which is running in different JVMs can be treated as sub agents. All these components/services are cascaded to the master agent. When any of the application component (subagent) comes up dynamically, the master agent can automatically discover it using the discovery service client MBean. Each subagent has Discovery Service (discovery packet responder) MBean installed with it. The protocol for communication between the master and subagents is easily configurable.

![Diagram of Distributed Application Management](image)

**Fig.5 Distributed Application Management**
Multi-Protocol Access For Existing SNMP Agent: The architecture for making existing SNMP agent accessible through other protocols like HTML, RMI, CORBA is given below. This protocol translation can be achieved with just a code generation setting in the JMX Compiler tool. The MIBs which the legacy SNMP agent supports must be loaded in the JMX Compiler and set the option for proxy in the JMX Compiler settings. The generated Model MBeans have the code generated for proxying the request received from any protocol adaptor, as SNMP request to the existing SNMP agent. This is achieved without writing a single line of instrumentation code.

Fig.6 Multi-Protocol Access for Existing SNMP Agent

SNMPv3 Security in Multi-Protocol Agent: AdventNet SNMP Adaptor supports complete SNMPv3. In a public network SNMPv3 protocol enables secured communication (using MD5, SHA and DES) between SNMP manager and Multi-Protocol master agent. Most of the time all the subagents will be running in the private network, hence the communication between the Multi-Protocol master agent and subagents will be in SNMPv1 or SNMPv2c protocol. The below architecture diagram describes this typical SNMP deployment for complex application/system.

Fig.7 SNMPV3 in Multi-Protocol Agent
Agent Toolkit Java Edition Overview

The AdventNet Agent Toolkit offers a comprehensive set of development and testing tools to cater the needs of Agent developers. The tools packaged with the product serve the agent developers right from defining the input (or) management information for the agent (a MIB or TL1 command set file) till the packaging and testing of the developed Agent. The Agents that can be developed using the tools provided in Agent Toolkit are the Multi-Protocol Agents, Standalone SNMP Agents and Standalone TL1 Agents.

Multi-Protocol Agents

Using Agent Toolkit the developers can enjoy a cradle to grave support in developing Multi-Protocol Agents for their applications. The steps performed in the development process of a Multi-Protocol Agent are:

- Defining the Input (management information) for an Agent.
- Generating code for the input.
- Instrumenting the code for a desired function.
- Compiling the generated Java files.
- Incremental code generation
- Packaging and Deploying the Agent and
- Testing the Agent developed.

Defining the Input

The first step towards agent development is the definition of the input. This input is framed based on the variables present in the application. Hence for defining the input, it is required to identify the variables that need to be managed. The input can be in the form of MBean interface definition, or could be a MIB definition. The MIB or Management Information Base, has a lot of syntaxes and for defining a MIB, prior knowledge about the syntaxes and ASN.1 notations is required. To learn in and out of the syntaxes in MIBs, is a tedious journey towards Agent development.

To achieve this, the AdventNet Toolkit provides an intuitive tool namely, the MIB Editor. Advanced knowledge on MIB is not required to run the tool. With some basic knowledge on the concepts used, it would be easy to get on with the MIB development.

Please go through the topic "Defining a MIB" available in Building SNMP agents section to know how to utilize MIB Editor for MIB creation. Note that MIB is one of the input or meta-data used for developing Multi-Protocol Agents.

Generating Code for the Input

JMX Compiler is a tool which converts MIB to a Java code (Model MBeans). In the generated code you can see getter and setter methods generated. For each ModelMBean, an XML file will be generated. This XML file has the descriptor for each Model MBeans (used to create ModelMBeanInfo objects) when any GET/SET requests comes from adaptor/connector client (for e.g. SNMP manager, HTML Browser, etc.,) it reaches the respective adaptor/connector which in turn will call the relevant MBean's getter/setter methods. Similarly when any notification is emitted by the MBean, it will be propagated to all the adaptors/connectors and they will convert them to their protocol specific (for examples, SNMPAdaptor converts notifications from MBeans to SNMP traps and sends to all the SNMP managers).
Instrumenting Code for a desired function

Instrumentation or editing of the generated code for a desired result is the major step that should be done in the agent developed. To explain the concept of instrumentation, let us consider that a System Administrator is remotely managing a mail-server. For this requirement (keeping track of the mails), if he had defined a MIB variable as "noOfMailsInQ", then the getNoOfMailsInQ() and setNoOfMailsInQ() methods get generated for that MIB. He just have to fill in the code for getting the number of mails in the queue, so that whenever a request is made from the manager to the agent, the agent will execute this method to return the value.

Compiling the source code

The code generated using the JMX Compiler should be compiled with the classes used by instrumentation in the classpath environment. The JMX Compiler allows to compile the generated agent by providing extensive environment/compiler options.

Incremental code generation

User may not wish to develop the complete agent at a single stretch and may go in for a phase by phase approach. In each phase they may wish to instrument/develop agent for portion of their MIB. This is supported in the JMX Compiler, where the user can generate agent and later regenerate for extra variables, in which case the newly generated code gets automatically merged with the already generated code, providing comprehensive code merging support.

Packaging and Deploying the Agent

The final step would be packaging and deploying the agent. The JMX Compiler provides option for deployment purpose ("Package Agent" option).

Please refer the topic "5.0 Configuring the Multi-Protocol Agent" available in Building Multi-Protocol Agents section, to know more about the options available in the JMX Compiler.

Testing the Agent

The generated Agent can be started in the JMX Compiler tool itself. The tool provides start/stop Agent options, thus providing the testing environment in the same tool. From the JMX Compiler tool, you can launch the manager/client applications (from right side bottom of the tool). These manager applications (browsers) namely: MBean Browser (RMI, CORBA, and HTTP client), MIB Browser (SNMP manager), TL1 Browser (TL1 manager) and HTML Browser (HTML client) can be used to test the various protocol adaptors/connectors of the Multi-Protocol Agent.

Please refer the topic, "6.0 Testing the Multi-Protocol Agent" available in Building Multi-Protocol Agents section, to learn more about the test tools.

Standalone SNMP Agents

Using Agent Toolkit, a developer can develop standalone SNMP Agent very easily. The standalone SNMP Agent supports SNMPv1/ SNMPv2c/ SNMPv3 versions completely along with proxy feature.

Now, let us walk through the steps involved in the development of the Standalone SNMP Agent:

Define a MIB using MIB Editor

The development of the SNMP Agent starts with defining a MIB. Before defining a MIB it is necessary to identify the objects/variables that has to be managed remotely. After identifying the objects to be managed, you can define a custom MIB for your application by using the MIB Editor tool which is used to create and modify SNMP MIB definitions. The output from the MIB Editor will be your MIB File.
Generate code for the MIB using MIB Compiler

Once you have the MIB file ready or if you already have a predefined MIB, the MIB Compiler will be the next tool to start with. The MIB Compiler takes your MIB as input and generates the necessary Java stub files based on the options you have provided. You can add your own code to the generated Java files to get the desired output from your application/device. This process is called instrumentation of the Agent. The generated and instrumented Java files are then compiled from the MIB Compiler to create the runtime Java SNMP Agent.

Test the Agent using MIB Browser

The Standalone SNMP Agent can be tested using the MIB Browser tool. This tool is used as the SNMP Manager to test the SNMP Agent. It takes the MIB as input, based on which a request is sent to the Standalone SNMP Agent running on a specified port. The Agent sends a response message to the manager based on the request received.

Standalone TL1 Agents

Using Agent Toolkit, a developer can develop standalone TL1 Agent conforming to Bellcore standards. Please refer to the topic Building TL1 agents to know more about Standalone TL1 Agents.

Now, let us walk through the steps involved in the development of the TL1 Agent:

Define the TCS (TL1 Command Set) file using TL1 Message Builder

The development of the STA starts with defining the TL1 Commands for your application. Before defining the TL1 commands, it is necessary to identify the objects/variables that have to be managed remotely. On identifying the objects, use the TL1MessageBuilder to create TL1 commands for each of the objects to be managed. These commands are stored in TL1 Message Syntax (TCS) file in the XML format.

Generate code for the TCS file using TL1 Compiler

The TL1 Compiler takes the TCS file as input and generates the necessary Java files, based on the options you have provided. You can add your own code to the generated Java files to get the desired output from your device/application. This process is called instrumentation of the Agent. The generated Java files are then compiled in the TL1 Compiler to create the runtime Java TL1 Agent.

Testing the Agent using TL1 Browser

The TL1 Browser, bundled with the Toolkit is used as the TL1 Manager to test the TL1 Agent. It takes the TCS file as input based on which a request is sent to the TL1 Agent. The TL1 Agent sends a response to the manager based on the request received.
Working with Editor Tools

The Editor tools bundled with Agent Toolkit are:

- MIB Editor
- TL1 Message Builder.

These tools are utilized for creating the input messages required for developing an agent. MIB (Management Information Base) are the input messages needed for a Multi-Protocol or SNMP Agent and TCS (TL1 Command Set) are the input messages needed for a TL1 Agent. Have a look at the explanation provided below that will help you work with the editor tools.

**MIB Editor**

Defining a MIB involves lot of syntax, conformance to SMIV1 and SMIV2 specifications and ASN.1 rules. Hence, creating a standard MIB File in a text editor, is a tedious process and usually leads to the following errors:

- Non-conformable to SMIV1 and SMIV2 specifications
- Non-conformable to ASN.1 rules
- Syntax errors

To overcome the above problem and to make agent development easy and simple, the toolkit provides an intuitive visual development tool, the MIB Editor. It is a tool used to create and modify SNMP MIB definitions. This tool has an easy-to-use wizard interface through which the various constructs conforming to SMIV1 and SMIV2 specifications and ASN.1 rules can be created. Advanced knowledge on MIB is not required to run this tool.

The following table gives you the input that goes into the MIB Editor, the output you receive from it and where the output is used in the agent development process:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Output Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object properties such as Object Type, name, syntax, access, status, description, textual convention etc.</td>
<td>MIB File</td>
<td>The MIB File is used in the development of Multi-Protocol and SNMP Agents.</td>
</tr>
</tbody>
</table>

Please refer the topic "4.0 Defining a MIB" available in Building SNMP Agents section, to know more about using the MIB Editor tool.

**TL1 Message Builder**

In TL1, every operation to be performed on a network element is defined as a TL1 message. The rules for defining a TL1 message is defined in GR-831 standard.

The TL1 Editor is an intuitive visual development tool used to create and modify TL1 Command Set and Data Set definitions in the XML format. TL1 Command Set (TCS) is a file containing definitions of management information so that a Network Element (NE) can be remotely monitored, configured and controlled. The TL1 Data Set definition file will have possible values for each input message defined in TCS.

The following table gives you the input that goes into the TL1MessageBuilder, the output you receive from it and where the output is used in the agent development process:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Output Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Code, Access Identifier Block, Message Payload Block etc.</td>
<td>TL1 Command Set File, TL1 Data Set File</td>
<td>The TL1 Command set and Data set files are used in the development of Standalone TL1 Agents</td>
</tr>
</tbody>
</table>

Please refer the topic "4.0 Defining a TCS File" available in Building TL1 Agents section, to know more about using the TL1 Message Builder.
Working with Compiler tools

Generating source code for a given MIB File or a TCS file, instrumenting the stub files and compiling the agent, forms the most important phase in the agent development process. To execute this process, the Agent Toolkit provides some intuitive visual development tools called the JMX Compiler, MIB Compiler and TL1 Compiler.

These tools take the input messages and generate java code for the same at ease. The steps involved in working with these tools are similar and thus have been listed commonly. They are:

- Generates java code for building Multi-Protocol, SNMP and TL1 Agents with the JMX Compiler, MIB Compiler and TL1 Compiler respectively.
- Instrument the generated code. This has to be done by the developer, based on his requirement. The toolkit provides a powerful UI, which allows the users to develop their agent applications specific to their requirements based on the options selected in the UI.
- Compile the generated code.
- Test the developed agent by invoking the test tools (MBean Browser, MIB Browser and TL1 Browser respectively.)

The following table gives you the input that goes into the compilers, the output received from it and exact place where the output is used in the agent development process:

<table>
<thead>
<tr>
<th>Compilers</th>
<th>Input</th>
<th>Output</th>
<th>Output Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>JMX Compiler</td>
<td>MIB file</td>
<td>Generated code</td>
<td>The generated code is instrumented and compiled to get the Multi-Protocol agent</td>
</tr>
<tr>
<td>MIB Compiler</td>
<td>MIB File</td>
<td>Generated Code</td>
<td>The generated code is instrumented and compiled to get the SNMP agent.</td>
</tr>
<tr>
<td>TL1 Compiler</td>
<td>TCS File</td>
<td>Generated Code</td>
<td>The generated code is instrumented and compiled to get the Standalone TL1 Agent</td>
</tr>
</tbody>
</table>
Working with Configuration Tools

V3 Configuration Tool

The SNMPv3 Agent Configuration tool is an intuitive tool used for configuring the USM and VACM details for SNMPv3 Agents.

The following table gives the input that goes into the V3 tool, the output received and the place where the output is utilized.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Output Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>User’s details and VACM details of the users are configured.</td>
<td>Configurations get stored in the snmpv3.ser file and is utilized when the v3 Agent is queried.</td>
<td>The configured users are queried from the Manager side. (MIB Browser).</td>
</tr>
</tbody>
</table>
Getting Started

Working with Management Tools

The management tools packaged with AdventNet Agent Toolkit are

- MIB Browser
- TL1 Browser
- MBean Browser

These browsers are used for testing the agents developed. An agent created by the compiler tools becomes a complete agent only when it responds properly to the queries made by the managers. Hence these management tools act as managers and tests the agents.

MIB Browser

MIB Browser is an intuitive visual test tool used in testing the Standalone SNMP Agent and Multi-Protocol SNMP Adaptor. The MIB Browser is a complete SNMP Browser that enables loading of MIB, MIB browsing, walking MIB tree, searching MIB and performing all other SNMP related functions to users. Viewing and operating the data available through a SNMP Agent on a managed device, e.g. a router, switch, hub etc., is made possible by using the MIB Browser.

Features of AdventNet MIB Browser

- The AdventNet MIB Browser provides the capability to load and view MIB Modules in a MIB Tree.
- The AdventNet MIB Browser facilitates in traversing the MIB tree to look at the definitions for each node of the MIB tree, to a particular object defined in the MIB.
- The AdventNet MIB Browser facilitates in performing the basic SNMP Operations like Get, GetNext, GetBulk and Set.
- The AdventNet MIB Browser supports the Multi-Varbind requests.
- The AdventNet MIB Browser enables real-time plotting of SNMP data in a Graph. Currently two types of graph are supported. They are line graph and bar graph.
- The line graph facilitates the viewing of multiple real-time graphs of data in the managed device as it changes in tune with time.
- The AdventNet MIB Browser provides a user-friendly view of SNMP Table data. The table data can be viewed in a separate window called SNMP Table Panel.
- Trap Viewer forms an integral part of the AdventNet MIB Browser, which enables the viewing of incoming traps. It also facilitates in parsing the traps.

The following table gives you the input that goes into the MIB Browser, the output you receive from it and where the output is used in the agent development process:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIB File, Request Type, OID, Value, Max-Repeaters, Non-Repeaters, Community etc.</td>
<td>Response from the Agent</td>
</tr>
</tbody>
</table>

TL1 Browser

The TL1 Browser is an intuitive visual test tool used for testing the Standalone TL1 Agent and Multi-Protocol TL1 Adaptor. The TL1 Browser enables loading of TL1 Command Set files and TL1 Data Set Files and performing all TL1 operations. It allows the user to view and operate on data available through a TL1 Agent on a managed device thereby managing the device. Example: Telecom switch.
Features of AdventNet TL1 Browser

- Capability to load and use multiple files having different set of input messages.
- Support to load either the TCS file or Text file in the TL1 tree
- Provides text area to display all the details of the input message like command code, TID, AID, Correlation Tag, General Block and Message Payload block.
- Provides text area to display response messages, autonomous messages and log messages.

The following table gives you the input that goes into the MIB Browser, the output you receive from it and where the output is used in the agent development process:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCS File, Command Code, Access Identifier, Message Payload Block etc.</td>
<td>Performs the required operation on the device/system</td>
</tr>
</tbody>
</table>

MBean Browser

MBean Browser is a tool provided by the Agent Toolkit in order to test the Multi-Protocol Agent for various RPC protocols. The user can view and operate on the data about the managed device available through the Multi-Protocol Agent. For accessing the data, the Multi-Protocol Agent has to be queried and this is done with the help of MBean Browser. Currently the protocols supported by MBean browser are:

- RMI
- CORBA
- HTTP

The MBean browser uses the client side API provided for accessing the MBeans through the connectors (RMI, HTTP or CORBA Connectors) at the server side. The default hostname and the ports are:

- CORBA - localhost: 1050
- RMI - localhost: 1099
- HTTP - localhost: 8050

The MBean Browser is extremely user friendly as all the functionalities are accessible with just a click of the mouse.
Agent Toolkit Java Edition Architecture

Multi-Protocol Agent Architecture

- Overview
- Instrumentation Level (MBeans)
- MBean Server
- Adaptors

Overview

AdventNet supports six adaptors/connectors for various protocols viz. SNMP, HTML, RMI, CORBA, TL1, and HTTP. Users can write their own adaptors for other protocols and plug them into the AdventNet Multi-Protocol Agent architecture. Since the JMX specification does not spell out any standards for adaptors, AdventNet's adaptors have proprietary implementation. However, if JMX evolves with specs for protocol adaptors, AdventNet's adaptors can be easily migrated to the same. The AdventNet Multi-Protocol agent implements all the specified multi-protocol interfaces and classes.

This chapter explains the AdventNet Multi-Protocol agent architecture, which comprises of the following components:

- Instrumentation level
- Agent Level (MBean Server)
- Connector level (Adaptors)
Instrumentation Level (MBeans)

The instrumentation level is the level in which the management data is collected by the agent or applied on the agent. This level is common for all the management protocols supported on the agent. The agent developer is required to implement the instrumentation at this level, and make it available as Java components called MBeans.

What are MBeans?

A Managed Bean or MBean in short is a Java object that implements specific interfaces and conforms to certain design patterns. MBeans encapsulate manageable objects as attributes and operations through their public methods and follow the design patterns for exposing them to management applications.

For example, a read-only attribute will have just a getter method, whereas a read-write attribute will have getter and setter methods.

Why MBeans?

Any object that is implemented as an MBean and registered with the agent can be managed from outside the agent's Java virtual machine. Such objects include the following:

- The resources your application wishes to manage.
- Value-added services provided to help manage resources.
- Components of the JMX infrastructure that can be managed.

Different Types of MBeans

AdventNet supports three types of MBeans namely:

- Standard MBean
- Dynamic MBean
- Model MBean

Each of these corresponds to different instrumentation requirements. The following section covers the three ways to instrument a resource by writing a standard MBean, by implementing a dynamic MBean, or by dynamically instantiating a configurable model MBean.

Standard MBean

Overview

A Standard MBean is the easiest way to define an MBean. It provides a static management interface, which is fixed at compile time and is invoked by reflection. The management information is exposed using the interface, which drives the standard MBeans.

The interface of a standard MBean is referred to as its MBean interface. Adding the suffix “MBean” to its class name forms its name. For example, the MBean interface name for the MBean called "MyClass" would be "MyClassMBean".

Standard MBean's Interface

The Standard MBean's interface is made up of the methods it makes available for reading and writing its attributes and for invoking its operations. When defining this management interface for the Java object, it should follow the design patterns listed in the instrumentation part of the JMX specification. The management interface of a standard MBean is composed of its attributes and operations.
• **Attributes**

Attributes are properties, which are exposed through the getter and setter methods in the MBean interface. Attributes may be read-only, write-only, or read-write. For any given attribute name, there can be at most one setter and one getter method, and if both are defined, they must conform to the data-type of the attribute.

• **Operations**

Operations are methods that can be remotely invoked on a resource by a management application. These are the methods other than the getter and setter methods present in the interface. They can be defined with any number of arguments of any type and can return value of any type.

The design patterns for operations are simple: any public method defined in the MBean interface that is not an attribute getter or setter is an operation. For this reason, getters and setters are usually declared first, so that all operations are grouped together. The name of an operation is the name of the corresponding method.

**Running the Standard MBean Example**

The Standard MBean example is available in the `<Agent Toolkit Home>/examples/jmx/standardMBean` directory. It illustrates the usage of the StandardMBean registered with the Multi-Protocol Agent. To run the example, please refer to the `Readme.html` file.

**Dynamic MBean**

**Overview**

Unlike the Standard MBean, the Dynamic MBeans do not have getter and setter methods for each attribute and operation. Instead, they have generic methods for getting or setting an attribute by name, and for invoking operations by name. These methods are common to all dynamic MBeans and are defined by the DynamicMBean interface.

The management interface is determined by the set of attribute and operation names to which these methods will respond. The getMBeanInfo method of the DynamicMBean interface returns a data structure, which describes the management interface. This metadata contains the attribute and operation names, their types, and the notifications that may be sent if the MBean is a broadcaster.

The management interface of a dynamic MBean is static, but this interface is exposed dynamically when the MBean server calls its `getMBeanInfo` method.

**When Dynamic MBean is used?**

Let's consider a scenario where the attributes and operations exposed are likely to change often. In this case, the instrumentation must provide more flexibility, such as being determined dynamically at run time. In such case, the standard MBean model doesn't work well, Dynamic MBeans bring this adaptability and provide an alternative instrumentation with more elaborate management capabilities.

**Note:** Ultimately, the implementation remains the same whether the instrumentation followed is a Standard MBean model or Dynamic MBean model. The difference is by the way the attributes and operations are exposed, i.e. whether by a static interface (Standard MBean model) or by runtime description (Dynamic MBean model).
The DynamicMBean Interface

The javax.management.DynamicMBean interface exposes the attributes and operations only at run time. Instead of exposing the attribute and operation names directly, it exposes them through relevant descriptors. The resource object's class, to be recognized as a dynamic MBean by the Multi-Protocol Agent or one of its super classes, must implement the DynamicMBean interface.

The DynamicMBean interface in the javax.management package is defined by the following methods:

```java
public interface DynamicMBean {
    public MBeanInfo getMBeanInfo();
    public Object getAttribute(String attribute) throws
            AttributeNotFoundException, MBeanException, ReflectionException;
    public void setAttribute(Attribute attribute) throws
            AttributeNotFoundException, InvalidAttributeValueException,
            MBeanException, ReflectionException;
    public AttributeList getAttributes(String[] attributes);  
    public AttributeList setAttributes(AttributeList attributes);
    public Object invoke(String actionName, Object params[], String
            signature[]) throws MBeanException, ReflectionException;
}
```

The `getMBeanInfo` method returns an MBeanInfo object, which contains the definition of the MBean's management interface. It contains a list of attribute names and types, a list of operations and their parameters, and other management information.

From the MBeanInfo object, the Multi-Protocol Agent can get the characteristics of all the Attributes/Operations. With this information of the Attribute, the Multi-Protocol Agent can access the attribute(s) value(s) through the `getAttribute(s)` methods. Similarly, calling the `setAttribute(s)` methods using the Dynamic MBean interface of the resource object can set attribute(s).

For invoking any method (the Operation that is exposed by the MBeanInfo) on the resource through a DynamicMBean object, the `invoke` method of the Dynamic MBean interface will be used by the Multi-Protocol Agent. If the operation is expected to return an object, then that object will be returned by this `invoke` method.

Running the Dynamic MBean Example

The Dynamic MBean example is available in the `<Agent Toolkit Home>/examples/jmx/dynamicMBean` directory. It illustrates the usage of the Dynamic MBean registered with the Multi-Protocol Agent. It would also serve to know how to develop a Dynamic MBean. To run the example, please refer to the Readme.html file.

Model MBean

Overview

The Model MBean specification is an interface that provides a management template for managed resources. It is a generic and configurable form of Dynamic MBean. The javax.management.RequiredModelMBean class is an implementation of Model MBean interface along with the Dynamic MBean interface. Model MBeans are used for instrumenting resources programmatically at run time.
To instrument a resource and expose it dynamically, follow the steps given below:

1. Instantiate the `javax.management.modelmbean.RequiredModelMBean` class in a Multi-Protocol Agent
2. Set the model MBean’s management interface
3. Designate the target object, which implements the management interface
4. Register the model MBean in the MBean server

**How Model MBean is different from other MBeans?**

In the case of Dynamic MBean model, the properties of the instrumentation class, which the user is interested to manage, are exposed through the Dynamic MBean interface implemented by this instrumentation class. In Model MBean model, these properties are described, i.e., initialized, by XML files.

The Model MBean information includes a descriptor for each attribute, constructor and notification in its management interface. A descriptor is an essential component of the Model MBean. It contains dynamic, extensible, and configurable behavior information for each MBean component. They provide the mapping between the attributes and the operations in the management interface and the actual methods that need to be called to satisfy the get, set, or invoke request.

**Merits of Model MBean**

The `javax.management.RequiredModelMBean` class acts as a generic template for creating manageable objects dynamically. Thus, resources, services, and applications can use this class by passing their instrumentation object and configure the exposure of default behavior through XML file or through DescriptorList and register it with the server (Multi-Protocol Agent).

For example,

```java
// User Instrumentation class
agentSystemInstrument = new AgentSystemInstrument();
// Initialize the RequiredModelMBean with the xml file.
rmm = new RequiredModelMBean
(com.adventnet.utils.jmx.Utilities.convertXmlToModelMBeanInfo
("./agentProjects/jmxProject/mySources/AgentSystemInstrument.xml"));
// the ObjectName for the instrumentation.
name = "AGENT_SAMPLE_MIB_JMX:type=AgentSystemInstrumentExt";
// finally register with the JMX Agent, i.e, MBeanServer.
server.registerMBean(rmm, new ObjectName(name));
```

**Key Features of RequiredModelMBean supported in this release:**

- XML file initialization support
- Flat file persistence support
- Cached values support

**XML Initialization**

**Why is it required?**

This allows the XML encoded data, representing the metadata of a Model MBean, to be used for initializing the MBean. The ModelMBean(Attribute)Info[], ModelMBean(Operation)Info[], andModelMBean(Notification)Info[] get initialized by this XML.
During initialization, the MBeanInfo of the Dynamic MBean interface gets created from the formed ModelMBean(Attribute)Info[], ModelMBean(Operation)Info[], ModelMBean(Notification)Info[].

Retrieval of various fields present in the XML (the descriptor for each attribute/operation /notification) class can be done as follows:

For AgentSystemInstrument.xml (the XML file for agentSystem group in agent-sample-mib.txt located in <Agent Toolkit Home>/mibs directory) file, the code for retrieving the OID for AgentDescr attribute from the Model MBean that is registered with the MBean Server with ObjectName "AGENT_SAMPLE_MIB_JMX:type=AgentSystemInstrument" is as follows:

```java
String name = "AGENT_SAMPLE_MIB_JMX:type=AgentSystemInstrument";
ModelMBeanAttributeInfo[] attrInfo = (ModelMBeanAttributeInfo[]) (server.getMBeanInfo(new ObjectName(name)).getAttributes());
Descriptor adss = attrInfo[0].getDescriptor(); // The Descriptor for AgentDescr attribute
ObjectName snmpValue = null;
try {
    snmpValue = new ObjectName(":" + (String) ((Descriptor) adss.getFieldValue("protocolMap")) .getFieldValue("snmp"));
} catch (MalformedObjectNameException mfe) {
    mfe.printStackTrace();
}
String oid = snmpValue.getKeyProperty("value"); // returns ".1.3.6.1.4.1.2162.4.1.1." // i.e the "value" key is inner tag for "protocolMap" for "snmp" protocol
```

- The following is the code to retrieve the description of the agent information.

```java
String descr = adss.getFieldValue("description"); //returns "Location information for the Agent"
```

- For AdiskTable.xml (the XML file for AdiskTable is located in the <Agent Toolkit Home>/jmxprojects/projectname/agent/bin/conf/xml/mbeans/ directory) file, the code for retrieving the name of the first column of the table is as follows:

```java
Descriptor adss = attrInfo[0].getDescriptor();
Descriptor columnDescriptor = (Descriptor) adss.getFieldValue("columnAttribute1");
String name = (String)
columnDescriptor.getFieldValue("name"); // return AdiskID
```

- The code for retrieving the SNMP syntax type of the column "AdiskName" is as follows:

```java
Descriptor descr = attrInfo[0].getDescriptor();
int count=0;
while(true) {
    if(descr.getFieldValue("columnAttribute"+ ++count)) == null) {
        count--;
        break;
    }
}
String type = null;
for(int j=0; j<count; j++) {
```
Descriptor columnDescr = (Descriptor) descr.getFieldValue("columnAttribute" + (j+1));
  if(columnDescr.getFieldValue("name").toString().equals("AdiskName")) {
    type = columnDescr.getFieldValue("type").toString();
    // This will give the type of AdiskName
    break;
  }

Similarly, you can get the operations/notifications field information from its relevant XMLString. The MBeanDescriptor.xml file follows the schema defined in DTD file located at <Agent Toolkit Home>/conf directory.

Flat File Persistence Support

The Model MBean attribute information is saved into a flat file depending on the persistence policy specified for each attribute in the XML file/descriptor.

Calling its setter method can change this default name. Thus, during the Multi-Protocol Agent's next startup, the Model MBean gets loaded with the stored attribute values in the constructor of RequiredModelMBean (Since, the RequiredModelMBean implements the PersistentMBean interface, which has load and store methods). The various Persist Policy supported are as follows:

- Never - the attribute is never persisted. This is useful for highly volatile data or data that has meaning only within the context of a session or execution period.
- OnTimer - the attribute is persisted whenever the persistPeriod service of the Model MBean expires.
- OnUpdate - the attribute is persisted every time the attribute is updated.
- NoMoreOftenThan - the attribute is persisted every time it is updated, unless the updates are closer together than the persistPeriod. This acts as an update throttling mechanism that helps to prevent, temporarily, highly volatile data from affecting the performance.

Caught Values Support

What is Caching?

Caching mechanism is provided to store attribute values that are offered by the management resource, in the Model MBean. Maintaining the values of fairly static variables in the Model MBean itself, allows it to return that value without contacting the managed resource.

The resource may also set its Model MBean to disable caching, implying that the resource would be called whenever an attribute is accessed. In this case, the managed resource is invoked and it returns the attribute values to the Model MBean. In turn, the Model MBean returns these value to the MBean Server, which returns them to the request originator, usually a management application.

What benefit does Caching provide?

In general, the adaptors access the application's RequiredModelMBean as the Multi-Protocol Agent returns it. If the data requested by the adaptor is current, then the managed resource is not interrupted with a data retrieval request. Therefore, direct interaction with the managed resource is not required for each interaction with the management system. This helps to minimize the impact of management activity on run-time application resources and performance.
The attribute descriptor contains currencyTimeLimit and lastUpdatedTimeStam fields. If the current time is past lastUpdatedTimeStam + currencyTimeLimit, then the attribute value is stale (or no value). If a getAttribute is received for an attribute with a stale value (or no value), then the getMethod for the attribute will be invoked and the returned value will be recorded for the attribute, lastUpdatedTimeStam will be reset, and the requester will be handed the new value.

If there is no getMethod defined, then the default value from MBeanInfo for the attribute will be returned.

**When to use Model MBean**

Model MBean usage is recommended over standard or dynamic MBeans on below scenario:

1. When you wish to use any protocol adaptors that has model:
   - If you wish to use SNMP adaptor, SOAP adaptor, CIM / WBEM adaptor (which has metadata model), the flexibility of defining the protocol properties (like OID, CIM schema) for an MBean attribute can be achieved only by ModelMBeans. On the other hand, if you wish to use protocol adaptors like RMI, HTTP (which doesn't have any meta model), then you can choose to use either standard or dynamic mbeans.
2. Out of the box persistence support for MBean and its attributes
3. Out of the box caching support for MBean and its attributes

**Running the Model MBean Example**

The Model MBean example is available in the `<Agent Toolkit Home>/examples/jmx/modelMBean` directory. It illustrates the usage of the Model MBean registered with the Multi-Protocol Agent. To run the example, please refer to the Readme.html file.

**Agent Level (MBean Server)**

The manager requests are received and processed at the agent level. This level handles the different protocols and converts manager requests into the common protocol independent instrumentation provided by MBeans. The task of converting from specific protocols is done by protocol-specific adaptors/connectors. These interface with a common MBean server, which controls and interacts with the MBeans.

The MBeanServer is the core component of the JMX infrastructure. It provides a protocol-independent and information model-independent framework with services for manipulating JMX manageable resources.

Registering a resource's MBean makes it visible to management applications and exposes it to management requests. The MBean server makes no distinction between the types of MBeans: standard, dynamic and model MBeans are all managed in exactly the same manner.

You can register objects in the MBean server through:

- The other objects in the agent application itself
- A remote management application (through a connector or a protocol adaptor)

The MBean server responds to the following management requests on registered MBeans:

- Listing and filtering MBeans
- Discovering and publicizing the management interface of MBeans
- Accessing MBean attributes for reading and writing
- Invoking operations defined in the management interface of MBeans
- Registering and deregistering Notification listeners for MBean notifications
Methods in MBean Server

The following are the methods for creating/registering/deleting MBeans:

- createMBean (creation)
- registerMBean (registration)
- unregisterMBean (deletion)

For accessing the attributes and operations on the MBean, the method names are same as specified in the DynamicMBean interface. The methods contain an extra parameter for passing the ObjectName information on the MBean, which you would like to access through the MBeanServer.

- getMBeanInfo
- getAttribute
- getAttributes
- setAttribute
- setAttributes
- invoke

For adding and removing NotificationListener for a NotificationBroadcaster MBean:

- addNotificationListener
- removeNotificationListener

For querying the MBeans based on a Query constraint along with a filter mechanism:

- queryMBeans
- queryNames

All management requests are handled by the MBean server, which dispatches them to the appropriate MBean. An MBean is identified by a unique symbolic name, called object name. The object name can be assigned either by the entity registering the MBean, or by the MBean itself, if its implementation has been designed to provide one. Managers give this object name to designate the target of their management requests.

Generally, whatever the access the API user would like to have on the MBean, it is achieved via the MBeanServer, with ObjectName of the MBean being passed as an extra parameter to all the methods.

Creating the MBean Server

The MBeanServer object can be created using the createMBeanServer static method in the MBeanServer factory class. This method creates an instance of MBeanServer implementation class, which is exposed as MBeanServer interface. This implementation class is present in the Agent Toolkit as MBeanServerImpl class. Thus, the following single line code will create an MBeanServer instance:

```java
MBeanServer server = MBeanServerFactory.createMBeanServer();
```
Connector Level (Adaptors)

Adaptors and connectors are protocol handlers, which expose MBeans information to their respective protocol clients/managers. The MBean (instrumentation) is common and needs to be instrumented only once. This instrumentation is made available through all the adaptors/ connectors.

An agent may contain any number of connectors or protocol adaptors, enabling it to be managed simultaneously by several managers, through different protocols. It is up to the agent application to coordinate all the port numbers on which it intends to receive requests.

The AdventNet Multi-Protocol Agent provides management of the Multi-Protocol Agent through the following adaptors. You can also create custom protocol adaptors and integrate them to the AdventNet adaptor framework. To do this, please refer to the topic "Writing custom protocol Adapters for the Multi-Protocol Agent".

- SNMP adaptor
- HTTP adaptor
- TL1 adaptor
- RMI adaptor
- CORBA adaptor
- HTML connector

SNMP Adaptor Architecture

SNMP Adaptor just acts as a wrapper class for AdventNet Standalone SNMP Agent, ensuring to provide the same SNMP agent stability that has come from the standalone SNMP agent.

![Fig.2 SNMP Adaptor Architecture](image-url)
CORBA Connector

The CORBA Connector enables CORBA clients to manage MBeans residing in AdventNet Multi-Protocol Agent. The CORBA connector uses CORBA naming service provided by the ORB implementation for CORBA object lookup. The IDL implemented by this CORBA connector is published with the toolkit, thus exposing the Agent Server remotely accessed by a non-Java connector client.

Fig 3. CORBA Connector Architecture
SNMP Agent Architecture

The AdventNet Standalone SNMP Agent Architecture supports all versions of SNMP and is designed in a manner to suit the requirements of Enterprise Vendors, Service Providers and the OEM markets. The following diagram gives an overview of the different components in SNMP Agent and their interactions between them.

Transport Provider

SNMP messages are sent over various transport protocols. The protocols currently supported by AdventNet SNMP Agents are UDP/IP and TCP/IP. The communication between a Manager and Agent takes place through these Transport Providers.

Message Processing Subsystem

The Message Processing Subsystem is responsible for sending and extracting data from received messages. The Message Processing Subsystem potentially contains multiple Message Processing Models. Each Message Processing Model defines the format of a particular version of an SNMP message and coordinates the preparation and extraction of each such version-specific message format.

In this level, code is generated for a MIB given as input. The Request Handlers generated takes care of processing the request from the Manager.

Thus, Message Processing Models receives the messages and processes them based on the version of the Message.
Instrumentation

The Message processed can be instrumented to return specific values of the application.

The generated code of the MIB Compiler has to be modified to get/set a MIB object value. Usually the MIB instrumentation differs based on the Agent implementation. Database, Runtime Memory and text file storage support is available to store MIB objects value.
TL1 Agent Architecture

This chapter explains the Standalone TL1 Agent architecture in the Agent Toolkit Java Edition.

The following architecture image explains the various modules in the Standalone TL1 agent.

**Transport Provider Interface**: Transport Provider is responsible for all communications between the Agent and the Manager. It uses the configured protocol for this purpose. It acts as a bridge between the Session layer and the underlying transport protocol. The advantage of using this approach is that the Session layer need not be aware of the underlying protocol used. For using a particular protocol, the user has to implement that protocol and plug in (or register) the protocol with the Transport Provider.

**TCP/IP**: TCP/IP is the default transport protocol for the TL1 Agent for managing remotely.

**Message Processing**: This module is responsible for processing the input message received from the client. This module is responsible for parsing and processing the input message received by the TL1 Engine. This module is also responsible for sending the output response messages and autonomous messages back to the NEs.

- **TL1 Parser**: TL1 Parser package is the default parser provided by this AdventNet TL1 Agent Toolkit. The parser is built as per the Bellcore standards. It can parse all TL1 messages like Input, Response, Autonomous, and Acknowledgement. Some options are also provided to set the parsing levels for each session. These methods can be called from the TL1Session. Apart from this, you can write your own parser and plug it into the TL1Session.
Acknowledgement Message Originator: TL1 acknowledgement messages will be sent to the managers, for instance, when the request processing takes time (In Progress acknowledgment message). This component in the TL1 adaptor facilitates to send all type of acknowledgement messages.

Output Formatter: It takes care of formatting and sending the response message to the TL1 Manager.

Autonomous Message Originator: TL1 agents are most preferred for fault management. And it becomes utmost important for efficiently sending autonomous messages. This component in TL1 adaptor is used to create and send the autonomous message to the TL1 Manager. The stub files can invoke methods in this component to send autonomous messages.

Built-in Commands: This module contains all the Built-in Commands that are implemented in the TL1 Agent. Built-in Commands are provisioning input commands common to all OS/NE interfaces. They are commonly used input commands that have already been implemented in the AdventNet TL1 Agent. Some general Built-in Commands as well as Built-in Commands for Delayed Activation have already been implemented in the TL1 Agent.

Security: This module contains the various Security Views associated with the Security implemented in the TL1 Agent. Security Validation such as User Authentication, Channel Authentication, Command Authentication, etc., takes place in the Security Module.

Instrumentation (TL1 Stubs Instrumentation)

The instrumentation files are the generated stub files for a given TL1 Command Set (.tcs) file. User or application/device-specific instrumentation can be done on these files.
AdventNet's Other Products

AdventNet, the Internet Management Solutions Company™ is the leading provider of open, scalable, extensible, and cross-platform management solutions for managing the Internet and e-commerce infrastructure. AdventNet's solutions ranges in scope from optical and core Internetworking Management Systems, Cable Modem, DSL, Storage, and Security Management to E-Commerce application management. In each of these fast-growing markets, AdventNet is the leading provider of technology.

AdventNet Solutions For Telecom Industry

Management APIs

AdventNet Agent Toolkit (C Edition) - AdventNet Agent Toolkit (C Edition) - a rapid tool for making prototypes and a development tool for building SNMP, HTTP and TL1 agents in strict ANSI C language. The Toolkit primarily focuses on System Management and Device Management and hence the run time agent is very modular, portable and customizable. The toolkit supports to provide multiple protocol (SNMP, HTTP and TL1) access to common management instrumentation, called as Multi-Protocol Agent. In addition to supporting Multi-Protocol agent, it also supports Standalone SNMP and TL1 agents.

AdventNet SNMP API - AdventNet SNMP API is a set of Java libraries (API) for creating cross platform Java and Web-based SNMP network management applets and applications. This package can be used to develop SNMP management applications and applets to manage SNMPv1, SNMPv2c and SNMPv3 agents (i.e. the management applications can be multi-lingual) and to talk to the agent systems using any of the three versions of the SNMP at the same time.

AdventNet TL1 API - It comprises a set of Java libraries for developers seeking to leverage the power of Java TM and other Internet technologies in quickly delivering Java and web-based solutions for managing TL1 devices. It provides a good base to build network management products and solutions for TL1 device management.

AdventNet CLI API - It consists of Low level, High level and Distributed APIs. The Low-level API consists of Java classes that implement the core CLI functionality including the communication framework, which enables the user to access the CLI API. The high-level APIs makes it easier the development of network management applications using the CLI libraries where as the Distributed functionality is carried out by RMI API.

AdventNet Mediation Server - A multi-protocol mediation service for telecom network management applications. It is designed for scalable multi-tier architectures such as J2EE that form the basis for many OSS and NMS products and solutions being built today. It provides a common interface based on XML messaging to applications using the Mediation Server.

Simulating Networks and Systems

AdventNet Simulation Toolkit - AdventNet Simulation Toolkit provides a comprehensive set of tools for creating a simulated environment consisting of networks, systems and applications. It not only supports setting up a simulated agent but also simulating an entire network on a Linux box. In addition to it, it also provides a rich set of utilities which help in setting up a simulated network in a very short time. All the real time conditions can be simulated very easily using Simulation Toolkit. It provides an intuitive GUI for configuration and comprehensive scripting capability in Java and JPython languages.

Testing

AdventNet Agent Tester - A powerful tool to test SNMP Agents. This product is intended for agent developers and QA teams to improve the quality and performance of their solutions, as well as for end users to verify compliance prior to deployment.
Other Utilities

AdventNet SNMP Utilities - Targeted towards end users, AdventNet SNMP Utilities is a set of cross-platform applications and applets for SNMP and Web based network management. These utilities can be used for device, element, application and system management. These tools can communicate and interact with any SNMP enabled device. The following tools form part of the product:

- **MIB Browser** - used to view and operate on data available through a SNMP agent in a managed device
- **Trap Browser** - used to parse and view the received traps
- **V3 Agent Configuration tool** - tool for configuring USM user tables and VACM entries for SNMPv3 agent
- **Proxy Forwarder** - used to translate SNMP V3 request for SNMP V1 and V2c agents.
- **WebServer and SAS Server** - facilitates in using SNMP management applets.

EMS, NMS, Provisioning & OSS Systems

AdventNet Web NMS - It is an open, massively scalable, carrier-grade management infrastructure platform built for the Internet age. It provides out-of-the-box application functions with tremendous flexibility to customize for a variety of domain specific needs.

AdventNet TMF, EMS and NMS - Based on Tele-Management Forum (TMF) standards, enable carriers to manage multi vendor/multi technology devices through a common management system that integrates across a diverse set of devices.

AdventNet V5 Protocol Stack - It is a portable, ANSI C implementation of the ETSI and ITU-T switching and signaling protocol stack for V5 interfaces.

AdventNet V5 Monitor - It is a PC based visual tool with an easy to use Java GUI for monitoring and analyzing ITU-T/ETSI V5 interfaces which is available for Linux and Windows OS.

AdventNet V5 LE Simulator - It is a PC based tool which simulates a V5 Local Exchange to test any V5 AN implementation and is available in both online and offline modes.

AdventNet Management Builder - It provides a comprehensive set of development tools to take care of the entire product life cycle needs of management applications, i.e., Design, Implementation, Testing Packaging, Deployment, Maintenance, and Enhancement.

AdventNet Solutions For Enterprises

AdventNet ManageEngine OpManager Remote Management - It is a scalable and secure solution for remotely monitoring and managing various resources (including network elements like routers, switches, applications like web-servers, database-servers etc. as well as system resources).

Managing J2EE Applications

AdventNet ManageEngine Applications Manager - It is a Real-Time, Standards-Based, Best-of-breed Solution offering a Scalable, Feature-rich, and Cost-effective Solution to manage all Components of the WebLogic Server and the associated application.

Testing Java/J2EE Applications

AdventNet QEngine - It is a comprehensive Cross-Platform Tool with a Compelling Business Value Proposition for Automating Large-scale Java and J2EE Application Testing.
Contacting AdventNet

AdventNet, the Internet Management Infrastructure Company™ is the leading provider of open, scalable, extensible, and cross-platform management solutions for managing the Internet and e-commerce infrastructure. AdventNet's solutions range in scope from optical and core internetworking management systems, cable modem, DSL, storage, security management to E-Commerce application management. In each of these fast-growing markets, AdventNet is the leading provider of technology.

AdventNet is eager to hear your comments, feedback, and suggestions that will go a long way in improving our products. We give you the contact information of our Corporate Headquarters, Sales, and Technical support teams. We also provide the link to subscribe to the Java network management mailing list.

During the evaluation phase, AdventNet provides support "at no cost". This support includes email access to our product specialists for problem resolution, clarifications in documentation and technical guidance. Feel free to send in your queries. We guarantee to respond as quickly as possible to make your evaluation a success.

Corporate Headquarters

<table>
<thead>
<tr>
<th>Web Site</th>
<th><a href="http://www.adventnet.com">http://www.adventnet.com</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>AdventNet Headquarters</td>
<td>AdventNet, Inc.</td>
</tr>
<tr>
<td></td>
<td>5645 Gibraltar Drive</td>
</tr>
<tr>
<td></td>
<td>Pleasanton, CA 94588 USA</td>
</tr>
<tr>
<td></td>
<td>Phone: +1-925-924-9500</td>
</tr>
<tr>
<td></td>
<td>Fax: +1-925-924-9600</td>
</tr>
<tr>
<td></td>
<td>Email: <a href="mailto:info@adventnet.com">info@adventnet.com</a></td>
</tr>
<tr>
<td>AdventNet Development Center</td>
<td>AdventNet Development Center (India) Private Limited</td>
</tr>
<tr>
<td></td>
<td>Sree Narayana Complex</td>
</tr>
<tr>
<td></td>
<td>11 Sarathy Nagar, Velachery, Chennai 600 042 INDIA</td>
</tr>
<tr>
<td></td>
<td>Phone: +91-44-243-2414</td>
</tr>
<tr>
<td></td>
<td>Email: <a href="mailto:info@adventnet.com">info@adventnet.com</a></td>
</tr>
</tbody>
</table>

Sales

For sales and product licensing enquiries, please use the contact information available in the following URL:
http://www.adventnet.com/contact.html

You can also send email to sales@adventnet.com at the AdventNet Headquarters main switchboard below.

Phone: +1-925-924-9500 and request for Sales
Fax: +1-925-924-9600

Technical Support

For support and bug-report, please mail to agent-support@adventnet.com. For those with support contracts, AdventNet provides priority support through email with a reply usually within 24 hours.
Calling AdventNet Support

You can call AdventNet Customer Response Center for logging and resolving any technical problems. This Center is operational 24 hours of the day staffed by experienced and knowledgeable product specialists. They are accessible between the hours of Monday 12.30 AM (PST) and Saturday 1.00 AM (PST) at:

++1 925 924 9500 (during the hours 8.30 AM PST and 6.00 PM PST)
++91 44 2436152 to 55

Please provide the following information while sending support mails.

- Release version of the product.
- Operating System and version
- JDK version
- Stack traces, if any
- Debug messages or hex dumps, if any
- CLASSPATH environment variable, if applicable

and any other information. This helps us provide faster responses to your query.

Feedback

We welcome your feedback that will help us improve the APIs and provide future enhancements. Send your feedback to agent-support@adventnet.com.
Installation Guide

- System Requirements
- Installation & Setup
- Starting Agent Toolkit
- Using Launcher
- Using License Manager
- Using Update Manager
- Using Tools
- Trouble Shooting at Startup
- FAQ of Installation and Startup
System Requirements

The following will be the minimum hardware and software requirements for running the Agent Toolkit Java Edition product.

Hardware Requirements

The performance seen with the AdventNet Agent Toolkit will depend considerably on the CPU and memory of both in the installed system. The following is the suggested minimum configuration:

<table>
<thead>
<tr>
<th>Operating Platform</th>
<th>Processor Type</th>
<th>Processor Speed</th>
<th>Memory</th>
<th>Hard Drive Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>Pentium III</td>
<td>500 MHZ</td>
<td>128 MB Ram or</td>
<td>75 MB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Higher</td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td>Pentium III</td>
<td>500 MHZ</td>
<td>128 MB Ram or</td>
<td>75 MB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Higher</td>
<td></td>
</tr>
<tr>
<td>Solaris</td>
<td>Ultra-5</td>
<td>400 MHZ</td>
<td>128 MB Ram or</td>
<td>75 MB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Higher</td>
<td></td>
</tr>
</tbody>
</table>

Software Requirements

<table>
<thead>
<tr>
<th>Platforms</th>
<th>Java Version</th>
<th>Web Server</th>
<th>XML parser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Family (NT, 2000, 2003)</td>
<td>JDK 1.2 &amp; above</td>
<td>Tomcat WebServer 3.2.4/ (Servlet 2.2 and jsp 1.1 compliant WebServer)</td>
<td>Jaxp 1.1</td>
</tr>
<tr>
<td>Linux 8.x</td>
<td>JDK 1.2 &amp; above</td>
<td>Tomcat WebServer 3.2.4/ (Servlet 2.2 and jsp 1.1 compliant WebServer)</td>
<td>Jaxp 1.1</td>
</tr>
<tr>
<td>Linux 6.2</td>
<td>JDK 1.2 &amp; above</td>
<td>Tomcat WebServer 3.2.4/ (Servlet 2.2 and jsp 1.1 compliant WebServer)</td>
<td>Jaxp 1.1</td>
</tr>
<tr>
<td>Linux 7.x</td>
<td>JDK 1.2 &amp; above</td>
<td>Tomcat WebServer 3.2.4/ (Servlet 2.2 and jsp 1.1 compliant WebServer)</td>
<td>Jaxp 1.1</td>
</tr>
<tr>
<td>Solaris 2.6</td>
<td>JDK 1.2 &amp; above</td>
<td>Tomcat WebServer 3.2.4/ (Servlet 2.2 and jsp 1.1 compliant WebServer)</td>
<td>Jaxp 1.1</td>
</tr>
<tr>
<td>Solaris 2.7</td>
<td>JDK 1.2 &amp; above</td>
<td>Tomcat WebServer 3.2.4/ (Servlet 2.2 and jsp 1.1 compliant WebServer)</td>
<td>Jaxp 1.1</td>
</tr>
<tr>
<td>Solaris 2.8</td>
<td>JDK 1.2 &amp; above</td>
<td>Tomcat WebServer 3.2.4/ (Servlet 2.2 and jsp 1.1 compliant WebServer)</td>
<td>Jaxp 1.1</td>
</tr>
</tbody>
</table>
Installation and Setup

Installing Agent Toolkit

AdventNet Agent Toolkit Java Edition can be downloaded from the site www.adventnet.com. This section discusses the installation procedure of AdventNet Agent Toolkit in various operating systems.

- **Installation on Windows NT and Windows 2000**: The AdventNet Agent Toolkit Java Edition is supplied in a self-extracting EXE format with an Install shield program for installation. Download the EXE file and run it to start the installation process and follow the instructions. The product is also available in zip version for Windows NT and Windows 2000. Download the zip file from www.adventnet.com and extract the files.

- **Installation on Linux/Solaris**: The AdventNet Agent Toolkit Java Edition is supplied as a zip file for Linux and Solaris. Users need to download the zip file from www.adventnet.com and extract the files.

Upgrading Trial User to Licensed User

The AdventNet Products can be downloaded from the web site for a specified evaluation period. Once the evaluation period is completed, the trial license will be expired. You can purchase the product from AdventNet by contacting the AdventNet sales Team. For any sales-related queries, send an e-mail to sales@adventnet.com.

Once you purchase the product, AdventNet will provide you a license file for permanent use of the product. Using License Manager section provides more information on this.

Installing a Service Pack

If in case of installing service packs (patch), please refer Update Manager section for more details.

Uninstalling the Product


2. Un-install the toolkit from the Control Panel using "Add/Remove programs" Option.
3. The un-installation procedure prompts for restarting the system.
4. Restart the system.

The above procedure will not remove files or entries generated after installation. The files or entries like logs need to be removed manually. Delete all the remaining files or entries and the AdventNet Agent Toolkit Java Edition X.X directory, to successfully un-install Agent Toolkit Java Edition (X.X denotes the release number).

Uninstalling the Agent Toolkit Java Edition in Linux/Solaris: Follow the steps given below to uninstall Agent Toolkit Java Edition in Linux/Solaris.

2. At the command prompt, type rm -rf AdventNet/JavaAgent, to successfully un-install Agent Toolkit Java Edition.
Starting Agent Toolkit

Starting the application in Windows 2000/NT

The installation process creates the AdventNet Agent Toolkit Program group in the Start Menu. The AdventNet Launcher can be started by choosing the "Start->Programs->Agent Toolkit Java Edition 6-> Agent Toolkit" icon from the Start menu. It helps to start all other Agent Toolkit applications bundled with this product. Thus the user can also start the JMX Compiler, MIB Editor, MIB Compiler by this method.

Starting the Application in Solaris / Linux

The AdventNet Launcher can be started by invoking the shell script Agent Toolkit.sh in the installed directory, it helps to start all other Agent Toolkit applications bundled with the product. Make sure that the environment variable "JAVA_HOME" is set before executing the shell script. The user can also start the MIB Editor, MIB Compiler and V3 Agent configuration tool as separate applications by invoking the shell scripts MibCompiler.sh, MibEditor.sh and v3apps.sh respectively.
Using License Manager

The main purpose of License Manager is to validate the License Key. AdventNet Agent Toolkit Java Edition software provides a valid license to run the product. The License Manager is integrated with all AdventNet products. This Manager is executed only once, that is, when the user runs any of the AdventNet's products for the first time.

There are two types of License for AdventNet's products:

- Trial User License
- Permanent License

The License Manager validates both of the above types.

Trial User Policy

AdventNet provides an Evaluation (Trial) period of 45 days for Agent Toolkit Java Edition. The evaluation license is a complete license that enables all the component features provided in AdventNet Agent Toolkit Java Edition.

Note: The evaluation period is from the date when Agent Toolkit Java Edition is downloaded and it doesn't take the installation date into account.

Registered User Policy

Agent Toolkit Java Edition provides four types of Developer license to registered users. The following table explains the license package structure and the protocols supported by each of the packages.

<table>
<thead>
<tr>
<th>Package Type</th>
<th>Supported Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Package</td>
<td>Multi-Protocol support with HTTP, HTML, RMI, and CORBA.</td>
</tr>
<tr>
<td>SNMP Package with Base Package</td>
<td>SNMP Adaptor, Standalone SNMP support, Base (Multi-Protocol support with HTTP, HTML, RMI, and CORBA)</td>
</tr>
<tr>
<td>TL1 Package with Base Package</td>
<td>TL1 Adaptor, Standalone TL1 support, Base (Multi-Protocol support with HTTP, HTML, RMI, and CORBA)</td>
</tr>
<tr>
<td>Complete Package</td>
<td>SNMP Adaptor, Standalone SNMP support, TL1 Adaptor, Standalone TL1 Agent, Multi-Protocol support with HTTP, HTML, RMI, and CORBA</td>
</tr>
</tbody>
</table>

To get the registered user license file, you must contact our sales team at sales@adventnet.com

Procedure for Licensing

After installing the product, you must provide the required licensing information for starting the product. The License Agreement will be invoked when you start the Application Launcher. The process begins by prompting the user to agree to the terms of the license agreement. To do so,
1. Select the checkbox labelled *I accept the License Agreement*. The Next button will be enabled.

2. Click Next to move to the next screen. There are 2 options displayed, viz., Evaluation User and Registered User.

3. If you want a trial user license, select Evaluation User and then click *Finish* to complete the process.

4. If you want a permanent user license, select Registered User. Click *Next* to move to the next screen.

5. Browse and select the Registered User License file in the screen for the license details.
6. Click Next. The next screen will display all the user names you had provided while requesting for trial user license and the company name.

7. Select one of the User Name from the ones displayed in the combo box.

8. Click Finish to complete the process.
To upgrade your license policy from Trial user to Registered user

- Start the Agent Toolkit Java Edition Launcher by running the Agent Toolkit.bat/sh file present under the <Agent Toolkit Home> directory.
- From the Help menu, select Register menu item. The License Manager Screen pops up.
- Use the above procedure to upgrade.

Please contact AdventNet for any clarifications about Licensing of the product.
Using Launcher

The AdventNet Agent Toolkit launcher provides a single UI from which the various tools can be invoked. Each tool can be started by double clicking on its icon or selecting the Options --> Run menu from the menu bar.

The application can also be run by right-clicking and selecting the "Run" option. The Agent Toolkit launcher can be toggled to the toolbar view by clicking the View --> Tool Bar menu from the menu bar. The launcher can be toggled back to the normal view by selecting the "normal view" icon.

The Launcher UI is organized as two panels - the top and the bottom panel. The top panel shows all the applications that are part of the AdventNet Agent Toolkit package. These tools are grouped according to the agent they help to develop such as SNMP Agent, TL1 Agent, Multi-Protocol Agent. All these applications can be started by double-clicking or by selecting the run option from the menu bar.

The bottom panel shows the currently running applications in the form of tabs. The applications can be terminated from the launcher itself by selecting Options --> Terminate menu or by right clicking and selecting the terminate option. The bottom panel also displays the error and exception messages of the respective applications.

Selecting the Options --> Settings menu pops up the Java Environment settings dialog box which can be used to set the various path and classpath variables needed to use the applications. Any new classpath settings, modification of the existing environment variables can also be done. The General settings are used to set the global settings which will be used throughout the AdventNet Agent Toolkit package.
Getting Started

Toolbar options
- **Settings**: This tool can be used for setting the options in Launcher.

- **Update manager**: This tool can be used for installing the Service Pack releases made by AdventNet Agent Toolkit Java Edition. Please refer to the Update manager which provides detailed explanation on how to install the Service Pack release.

- **Developer Support**: This tool leads to the Contacting AdventNet page of Product Help documentation.

- **Log**: Using this tool you can keep track of the error messages stored in log files.

- **Exit**: Exit tool can be used for exiting from the Agent Toolkit application.

Menu bar options in Launcher
- **Options**: These menus can be used for setting the options in launcher.

- **Run**: Normal-editing functions can be performed here.

- **View**: To view the launcher UI in normal view and tools view.

- **Help**: link to product help documents.

Setting Options in Launcher

The following environment settings can be set using the settings option in the launcher:

- **Browser path** - path for the Web browser. The Web browser is used for loading the applets and for viewing the help files. To set the path, click on the browse button and select the corresponding executable for the browser. (eg. `c:\Program Files\communicator\netscape.exe` or `user/bin/netscape`). Please note that the complete file name has to be given, not just the path where the executable lies.

- **JAVA HOME** - sets the JAVA_HOME environment variable. This is used for running the applications.

- **JAVA Classpath** - the classpath settings necessary for running the application is set here. By default the necessary classpaths are already set. Additional classpath can be added to the existing list. Please note that the classpath settings take precedence in the order in which they are listed.

The Applications folder in the settings panel displays all the applications that are part of the AdventNet Agent Toolkit package. Any property specific to the application can be set by selecting the corresponding application. The settings tab corresponding to each application displays the class path settings specific to the application. Any new classpath settings can be added using this option.
Using Update Manager

The Update Manager is a tool which is used for installing the service packs (.ppm file) over a product. The service pack may contain certain bug fixes and new feature additions for a particular release version of the product. The service pack can be classified to be either an ordinary without referring to any context of the product (Non - Context based) or it may be context based. This document explains about how to use the Update manager for installation of Software upgrades in AdventNet products. It can be also used by products developed over these products.

Context based patch implies that the patch will have the consolidated upgrades for all the different components of the same product. The same patch can be used for updating any context of the product. The Update manager also provides a feature for having Mandatory context and Optional contexts. The Mandatory context can also be called compulsory context and will not be displayed at the time of installation. An Optional context is some additional upgrades, but is not absolutely necessary for the product to function.

The Update Manager also has some useful validation incorporated. This validation includes compatibility checks. If a service pack is not compatible i.e., service pack of one product is installed in other product, or service pack of one product version installed in other version, then update manager will not allow the user to install these types of service packs.

To open the Update Manager tool run the UpdateManager.bat/sh file present in the <Agent Toolkit Home>/bin directory. It can also be opened using launcher by running Agent Toolkit.bat file available at <Agent Toolkit Home> directory where the Update Manager can be chosen by selecting Options > Update Manager or choosing the icon Update Manager from the tools bar. The Update manager UI lists the different versions of service pack installed for this product. The details of each of the service packs can be viewed by selecting the corresponding item in the list and clicking the details button.
The Update manager can be operated in two modes i.e., UI and non-UI mode. For starting the update manager in command line or non-UI mode, there are two ways.

1. Run the UpdateManager.bat/sh file with -c as a command line argument.
2. The user has to edit the script file ie., UpdateManager.bat or UpdateManager.sh depending on the OS. The change required is very simple, just an addition of one more argument -c (%JAVA_HOME%\bin\java -Xmx100m com.adventnet.tools.update.installer.UpdateManager -u conf -c).

Update Manager UI

The main features of the Update Manager UI are:
- To install the patch.
- To view Installed Service pack details
- To Uninstall a installed Service pack.

To Install the Service Pack

During the installation of a patch, the corresponding files related to bug fixes are updated inside the product installation directory according to the respective file locations.
- Click the Install button to start installation procedure. The installation wizard will pop up.
- Use the browse button to give the service pack file (.ppm). Once this file is given, the View Readme button will get enabled. Only compatible service pack file will be opened.
- On clicking the View Readme button the readme file related to this service pack is displayed in a separate window.
- On clicking the next button the various optional contexts available are displayed for selection.
  The user can select the context required and click install button.
- When install button is clicked a new panel is displayed giving details of service patch and progress in installation.

The tool will intimate the user on completion of the installation process.
To View Installed Service Pack Details

The installed service packs are listed in the main window of the Update manager.

- The details can be viewed by selecting the service pack version and clicking the Details button. The following screen will be shown.

A dialog pops-up displaying the General details about the patch in a tab. It contains the patch name, description, installed size and installation time.
- The installation log messages are shown below these details.
- The other tab displays the README related to this Service pack.
- The details can also be viewed by double clicking the required service pack version.
To Uninstall a Service Pack

To uninstall the service pack:

- Select the version from the list shown. Then click on the Uninstall button. The following screen will be shown.

- This will bring up a dialog, which shows the service pack to be uninstalled. It also lists all the related service packs that will be uninstalled.
- On clicking the Uninstall button the status screen is displayed, which shows the progress in uninstallation.

Update Manager in Command Line

The main features of the Update Manager command line are:

- To install the patch through command line.
- To uninstall a service pack using command line.

To Install the Patch through Command Line

- On invoking the update manager in command line mode, it will ask the user to select an option.
- For installation enter i and press enter.
- The update manager will prompt for Service pack file name. Here enter the full path of the .ppm file.
- If Optional contexts are available in the Service pack the Update manager will ask the user whether he/she wants to install them. On selecting to install the optional context, the list of contexts is printed in the console. Now one can select the contexts required.

Now the Update Manager will start the installation process.
To Uninstall a Service Pack using Command Line

- On invoking the update manager in command line mode, it will ask the user to select an option.
- For installation enter u and press enter.
- The tool will request the user to select the version to Uninstall. Here the user will have to enter the serial number of the version to be uninstalled.

Now the Update Manager will start the Uninstallation process.
Getting Started with Tools

MIB Editor

MIBs are specifications containing definitions of management information so that networked systems can be remotely monitored, configured and controlled. The rules for writing MIBs are defined in a collection of documents called the Structure of Management Information (SMI).

The MIB Editor is used to create and modify SNMP MIB definitions. The MIB definition and modification is most useful for enterprise MIB creation and prototyping. The MIB Editor has an easy-to-use wizard interface through which the various constructs conforming to SMIV1 and V2 specifications and ASN.1 rules can be created.

Starting MIB Editor

The MIB Editor can be started in any of the following ways:

- From the Agent Toolkit launcher, by running the `<Agent Toolkit Home>/AgentToolkit.bat` for windows OS (or) `<Agent Toolkit Home>/AgentToolkit.sh` for other OS.
- From the bin directory, by running `<Agent Toolkit Home>/bin/MibEditor.bat` for windows OS (or) `<Agent Toolkit Home>/bin/MibEditor.sh` for other OS.
- Yet another way to invoke the MIB Editor in windows environment is by clicking Start menu -> Programs -> Agent Toolkit Java Edition 6 -> SNMP Agent -> MIB Editor.

Running the file, by default will take the JAVA_HOME path as set in the PATH variable. If the JAVA_HOME path is not set properly, it is essential that you set the path before proceeding further. You can do this by editing the setenv.bat/setenv.sh file in the bin directory.

MIB Compiler

The MIB Compiler is a tool, which provides an integrated environment to generate java source code for developing a standalone SNMP Agent. It takes the MIB file as input and based on the MIB generates the necessary java files. It also allows you to instrument the code and compile the Agent.

Starting MIB Compiler

The MIB Compiler can be started in any of the following ways:

- From the Agent Toolkit launcher, by running the `<Agent Toolkit Home>/AgentToolkit.bat` for windows OS (or) `<Agent Toolkit Home>/AgentToolkit.sh` for other OS.
- From the bin directory, by running `<Agent Toolkit Home>/bin/MibCompiler.bat` for windows OS (or) `<Agent Toolkit Home>/bin/MibCompiler.sh` for other OS.
- Yet another way to invoke the MIB Compiler in windows environment is by clicking Start menu -> Programs -> Agent Toolkit Java Edition 6 -> SNMP Agent -> MIB Compiler.

Running the file, by default will take the JAVA_HOME path as set in the PATH variable. If the JAVA_HOME path is not set properly, it is essential that you set the path before proceeding further. You can do this by editing the setenv.bat/setenv.sh file in the bin directory.

MIB Browser

The AdventNet MIB Browser is a complete SNMP MIB Browser that enables loading of MIB's, MIB Browsing, walking MIB tree, searching MIB's and performing all other SNMP related functions to users. Viewing and operating the data available through an SNMP agent on a managed device, e.g. a router, switch, hub etc., is made possible by using the MIB Browser.
Starting MIB Browser

The MIB Browser can be started in any of the following ways:

- From the Agent Toolkit launcher, by running the `<Agent Toolkit Home>/AgentToolkit.bat` for windows OS (or) `<Agent Toolkit Home>/AgentToolkit.sh` for other OS.
- From the bin directory, by running `<Agent Toolkit Home>/bin/MibBrowser.bat` for windows OS (or) `<Agent Toolkit Home>/MibBrowser.sh` for other OS.
- Yet another way to invoke the MIB Browser in windows environment is by clicking Start menu -> Programs -> Agent Toolkit Java Edition 6 -> SNMP Agent -> MIB Browser.

Running the file, by default will take the JAVA_HOME path as set in the PATH variable. If the JAVA_HOME path is not set properly, it is essential that you set the path before proceeding further. You can do this by editing the setenv.bat/setenv.sh file in the bin directory.

SNMP V3Admin

The SNMP V3Admin is a tool which is used for configuring USM user tables and VACM tables for SNMPv3 agent at runtime.

Starting SNMP V3Admin

The SNMP V3Admin can be started in any of the following ways:

- From the Agent Toolkit Launcher, by double clicking on SNMP V3Admin icon under SNMP agent.
- From the bin directory, by running `<Agent Toolkit Home>/bin/SnmpV3AdminTool.bat>` for windows OS (or) `<Agent Toolkit Home>/bin/SnmpV3AdminTool.sh>` for other OS.
- Yet another way to invoke the MIB Browser in windows environment is by clicking Start menu -> Programs -> Agent Toolkit Java Edition 6 -> SNMP Agent -> SNMP V3Admin.

Running the file, by default will take the JAVA_HOME path as set in the PATH variable. If the JAVA_HOME path is not set properly, it is essential that you set the path before proceeding further. You can do this by editing the setenv.bat/setenv.sh file in the bin directory.

JMX Compiler

The JMX Compiler is a tool, which provides an integrated environment to generate java source code for developing a Multi-Protocol agent. It takes the MIB file as input and based on the MIB generates the necessary java files. The generated code will be based on the Model MBeans specified by JMX specification.

Starting JMX Compiler

The JMX Compiler can be started in any of the following ways:

- From the Agent Toolkit launcher, by running the `<Agent Toolkit Home>/AgentToolkit.bat` for windows OS (or) `<Agent Toolkit Home>/AgentToolkit.sh` for other OS.
- From the bin directory, by running `<Agent Toolkit Home>/bin/JmxCompiler.bat` for windows OS (or) `<Agent Toolkit Home>/bin/JmxCompiler.sh` for other OS.
- Yet another way to invoke the JMX Compiler in windows environment is by clicking Start menu -> Programs -> Agent Toolkit Java Edition 6 -> Multi-Protocol Agent -> JMX Compiler.

Running the file, by default will take the JAVA_HOME path as set in the PATH variable. If the JAVA_HOME path is not set properly, it is essential that you set the path before proceeding further. You can do this by editing the setenv.bat/setenv.sh file in the bin directory.
MBean Browser

The AdventNet MBean Browser is a Manager application which enables the user to access a Multi-Protocol agent's MBeans and use the provided services through the RMI / CORBA/ HTTP protocol adaptor or connector. To accomplish this, the Multi-protocol agent must be started which instantiates the different adaptors/connectors at their specified port numbers.

The MBean Browser uses the client side API provided for accessing the MBeans through the connectors (RMI Adaptor, CORBA Adaptor, or HTTP Connector) at the server side.

Starting MBean Browser

The MIB Browser can be started in any of the following ways:

- From the Agent Toolkit launcher, by running the `<Agent Toolkit Home>/AgentToolkit.bat` for windows OS (or) `<Agent Toolkit Home>/AgentToolkit.sh` for other OS.
- From the bin directory, by running `<Agent Toolkit Home>/bin/MBeanBrowser.bat` for windows OS (or) `<Agent Toolkit Home>/MBeanBrowser.sh` for other OS.
- Yet another way to invoke the MIB Browser in windows environment is by clicking Start menu -> Programs -> Agent Toolkit Java Edition 6 -> Multi-Protocol Agent -> MBean Browser.

Running the file, by default will take the JAVA_HOME path as set in the PATH variable. If the JAVA_HOME path is not set properly, it is essential that you set the path before proceeding further. You can do this by editing the setenv.bat/setenv.sh file in the bin directory.

TL1 Message Builder

TCS is a file containing definitions of management information so that a network element (NE) can be remotely monitored, configured and controlled. The rules for defining a TL1 message are defined in GR-831 standard. The TL1 Message Builder is an intuitive GUI tool that helps you create and modify TL1 Command Set and Data Set definitions in the XML format. The TL1 Data Set definition file will have possible values for each command defined in TCS.

Starting TL1 Message Builder

The TL1 Message Builder can be started in any of the following ways:

- From the Agent Toolkit launcher, by running the `<Agent Toolkit Home>/AgentToolkit.bat` for windows OS (or) `<Agent Toolkit Home>/AgentToolkit.sh` for other OS.
- From the bin directory, by running `<Agent Toolkit Home>/bin/TL1MessageBuilder.bat` for windows OS (or) `<Agent Toolkit Home>/bin/TL1MessageBuilder.sh` for other OS.
- Yet another way to invoke the TL1 Message Builder in windows environment is by clicking Start menu -> Programs -> Agent Toolkit Java Edition 6 -> TL1 Agent -> TL1 Message Builder.

Running the file, by default will take the JAVA_HOME path as set in the PATH variable. If the JAVA_HOME path is not set properly, it is essential that you set the path before proceeding further. You can do this by editing the setenv.bat/setenv.sh file in the bin directory.

TL1 Compiler

The TL1 Compiler is a tool, which provides an integrated environment to generate java source code for a standalone TL1 Agent. It takes the TL1 command set (TCS) file as input and generates code based on the commands defined in the TCS file.
Staring TL1 Compiler

The TL1 Compiler can be started in any of the following ways:

- From the Agent Toolkit launcher, by running the `<Agent Toolkit Home>/AgentToolkit.bat` for windows OS (or) `<Agent Toolkit Home>/AgentToolkit.sh` for other OS.
- From the bin directory, by running `<Agent Toolkit Home>/bin/TL1Compiler.bat` for windows OS (or) `<Agent Toolkit Home>/bin/TL1Compiler.sh` for other OS.
- Yet another way to invoke the TL1Compiler in windows environment is by clicking Start menu -> Programs -> Agent Toolkit Java Edition 6 -> TL1 Agent -> TL1Compiler.

Running the file, by default will take the JAVA_HOME path as set in the PATH variable. If the JAVA_HOME path is not set properly, it is essential that you set the path before proceeding further. You can do this by editing the setenv.bat/setenv.sh file in the bin directory.

TL1 Browser

AdventNet TL1 Browser is a full-fledged GUI application that can be used to query a TL1 device and get responses. It allows the user to view and operate on data available through a TL1 agent on a managed device thereby managing the device. Example: Telecom switch.

Starting TL1 Browser

The TL1 Browser can be started in any of the following ways:

- From the Agent Toolkit launcher, by running the `<Agent Toolkit Home>/AgentToolkit.bat` for windows OS (or) `<Agent Toolkit Home>/AgentToolkit.sh` for other OS.
- From the bin directory, by running `<Agent Toolkit Home>/bin/TL1Browser.bat` for windows OS (or) `<Agent Toolkit Home>/bin/TL1Browser.sh` for other OS.
- Yet another way to invoke the MIB Editor in windows environment is by clicking Start menu -> Programs -> Agent Toolkit Java Edition 6 -> TL1 Agent -> TL1 Browser.

Running the file, by default will take the JAVA_HOME path as set in the PATH variable. If the JAVA_HOME path is not set properly, it is essential that you set the path before proceeding further. You can do this by editing the setenv.bat/setenv.sh file in the bin directory.
**Trouble Shooting at Startup**

**Out of environment space**
- **Reason:** This error message is thrown while starting batch files present in the `<Agent Toolkit Home>/bin` directory. Problem with the Windows’ environment variable settings causes this error.
- **Solution:** Modify the properties of your DOS prompt (right click on the title bar of the DOS prompt window, select properties, choose memory tab, in that increase the value of the initial environment field (preferably 4096)).

**Too many parameters**
- **Reason:** This error arises due to problem with the Windows' PATH variable which doesn't accept paths which include space in their names (e.g., c:\program files)
- **Solution:** Modify the path variable to a DOS compatible one (i.e., c:\program files can be changed to c:\progra~1).

**MS-Dos window starts and closes immediately**
- **Reason:** While starting a batch file the MS-Dos window appears and is closed immediately. This occurs when JAVA_HOME is not set correctly in the setenv.bat file present in the `<Agent Toolkit Home>/bin` directory.
- **Solution:** Please verify if JAVA_HOME is set correctly in the setenv.bat file. The jdk version and the drive in which you have installed jdk should be the same as that specified in the file. If not, modify the JAVA_HOME variable and save the file and restart the tool.

**Graphics configuration error**
- **Reason:** This problem occurs if the monitor is configured for 16 colors mode. Swing /awt components of JDK requires at least 256 colors to run.
- **Solution:** Change the monitor configurations to 256 colors or install the Agent Toolkit in a monitor which supports at least 256 colors.

**javac not recognized as an internal or external command**
- **Reason:** The PATH variable is not set properly.
- **Solution:** Set the PATH variable if you want to be able to conveniently run the SDK executables (javac.exe, java.exe, javadoc.exe, etc.) from any directory without having to type the full path of the command. If you don't set the PATH variable, you need to specify the full path to the executable every time you run it, such as: `C:\jdk1.2.2\bin\javac MyClass.java`. It's useful to set the PATH permanently so it will persist after rebooting. To set the PATH permanently, add the full path of the jdk1.2.2\bin directory to the PATH variable. Typically this full path looks something like `C:\jdk1.2.2\bin`. Set the PATH as follows, according to whether you are on Windows NT or Windows 95/98.
Windows NT - To set the PATH permanently:

1. Start the Control Panel, select System, select Environment, and look for "Path" in the User Variables and System Variables. If you're not sure where to add the path, add it to the right end of the "Path" in the User Variables. A typical value for PATH is: \C:\jdk1.2.2\bin

2. Click "Set", "OK" or "Apply".

3. The new path takes effect in each new Command Prompt window you open after setting the PATH variable.

Windows 98 and Windows 95 - To set the PATH permanently:

1. Open the AUTOEXEC.BAT file and add or change the PATH statement as follows:

2. Start the system editor. Choose "Start", "Run" and enter sysedit, then click OK. The system editor starts up with several windows showing. Go to the window that is displaying AUTOEXEC.BAT.

3. Look for the PATH statement. (If you don't have one, add one.) If you're not sure where to add the path, add it to the right end of the PATH. For example, in the following PATH statement, we have added the bin directory at the right end:

   PATH C:\WINDOWS; C:\WINDOWS\COMMAND;C:\JDK1.2.2\BIN

4. To make the path take effect in the current Command Prompt window, execute the following:
   \C:> c:\autoexec.bat

To find out the current value of your PATH, to see if it took effect, at the command prompt, type: \C:> path

java.lang.NullPointerException --
com.adventnet.tools.license.LicensingKey.getDays

- **Reason:** The exception has been noticed with jdk1.4 or J2sdk1.4.0 if Agent Toolkit installation PATH has got space in their directory names. This is caused because of a bug in jdk1.4 where getting a resource (a file here) is not possible under this circumstance.

- **Solution:** Please install the Toolkit in a PATH, which does not have space and try running the application.
FAQ - Installation and Startup

1. What is an Agent?

Network management system contains two primary elements: a manager and agents. The Manager is the console through which the network administrator performs network management functions. Agents are the entities that interface to the actual device being managed. Bridges, Hubs, Routers or network servers are examples of managed devices that contain managed objects.

These managed objects might be hardware, configuration parameters, performance statistics, and so on, that directly relate to the current operation of the device in question.

2. What are the tools provided by AdventNet Agent Toolkit? Can each tool function as a separate application?

The tools provided by AdventNet Agent Toolkit are as follows:

- MIB Editor: This tool is used to create, edit and instrument MIB files.
• JMX Compiler: This tool generates stub files for building Multi-Protocol Agent taking the MIB file as input.
• MIB Compiler: This tool generates stub files for building SNMP agents taking the MIB file as input.
• SNMP V3 Agent Configuration Tool: This tool can be used for configuring USM user tables and VACM entries for SNMPV3 agent
• MIB Browser: This tool is used to view and operate on data available through an SNMP agent in a managed device.
• MBean Browser: This tool is used for testing JMX compliant agent using various protocols such as RMI, CORBA and HTTP.
• TL1 Browser: This tool allows the user to view and operate on data available through a TL1 agent in a managed device.
• TL1 Message Builder: This tool can be used for defining the TL1CommandSet (.tcs file) for a TL1 agent.
• TL1 Compiler: This tool that can be used to develop a TL1 agent. It takes the tcs file as input and generates code for it to form a TL1 agent.

Yes. Each tool can function as a separate application. You can use the .bat/.sh file to load each application separately.

3. What are the different types of agent developed using Agent Toolkit?

The AdventNet Agent Toolkit allows you to develop SNMP Agent, JMX Compliant agent and a TL1 agent.

4. What are the basic steps involved in developing an agent?

The basic steps involved in developing an agent are as follows:
• Define the MIB/TCS file using MIB Editor/TL1 Message Builder.
• Load the MIB/TCS file defined or an existing MIB/TCS file into the MIB Compiler/JMX Compiler/TL1 Compiler.
• Generate source files by choosing the appropriate options in the Compiler.
• Instrument the Stub files.
• Compile stub files.
• Start Agent.

5. What Operating systems does it run on?

Since the product is developed in Java it is platform independent. You can use the product in any Operating System with the JDK port of that particular OS. On our part we have tested the product in Windows95, Windows98, Windows NT, Windows 2000, Linux and Solaris.

6. What are the areas of Management where the Agent Toolkit Java Edition agent is best suited?

This product is well suited for the following market segments:
• Application management
• OEM vendors
• Enterprises
• System management
• Middleware ISV’s
• Service Providers

AdventNet Inc.
7. What are the Hardware and Software requirements for running the Agent Toolkit Java Edition?

To know about the minimum hardware and software requirements for running the Toolkit, please refer to the topic System requirements.

8. What are the technologies used by the Agent Toolkit Java Edition?

The technologies used by the Agent Toolkit Java Edition are:

- JMX
- XML
- SNMP
- HTTP
- TL1

To know more about each of these technologies, please refer to the Technology Overview section.

9. What are the JDK versions supported by the Agent Toolkit Java Edition?

The toolkit supports the following JDK version:

- Runtime Agent: Version 1.1.6 and higher.
- Tools in Agent Toolkit Version 1.2 and higher.

Note: Tools have been tested in SUN JDK's from JDK1.2 to JDK 1.4

10. How do I choose which type of Agent is best suited for my application?

The Agent which is best suited for your application depends upon the protocol support which you want to provide for the management of the agent. Please refer to the topic Agent Toolkit Java Edition Architecture, to know more.

11. What are the standards to which the Agent Toolkit complies?

Please refer to the topic Standards Compliance in each Building the Agent section to know about the standards supported by the toolkit.

12. What third party tools are bundled with your product?

Please refer to the topic Third party tools to know about the Third party tools bundled with the product.

13. Does your agent support Thread? Or is your agent thread safe?

Yes, the java agent supports multi-threaded processing and is thread safe.

14. Does your agent supports database?

Yes, the toolkit provides database support for storing the data and accessing this database table. For more information on Database support, please refer to the topic Supported Storage Types.

15. Is there any tool bundled with your product, to install a patch? Does the latest patch include the bug fixes done in previous patches?

The tool "Update Manager", bundled with the toolkit, is used for updating patches/service packs to the base version of the product. Please refer the topic Update Manger to know more about using the tool.

16. Is there any difference between the evaluation version and the permanent version of the Java Agent Toolkit?

No, the evaluation version and permanent version are the same.
17. Is code merging supported in your toolkit?

Yes, code merging is supported to merge two different versions of java files. Please refer the topic Code Merging to know more about this feature.

18. I know that AdventNet has both C and Java products. When would I, as a developer of an SNMP agent for a Network Interface Card (NIC), choose one over the other?

The C agent has less footprint, will execute faster and the end customer will not be required to install a JVM on their server to use the agent, but has some porting issues. The Java agent needs JVM on the target platform but has less porting issues.

Architecture wise, both AdventNet C agent and Java agent follow the same modular approach. So the choice will be based on where the agent is going to reside. For example, if the agent runs on Linux and makes JNI calls to NIC, then Java could be preferred. If the agent is intend to run in the card, then JVM could not be affordable, hence C agent.

19. Using Agent Toolkit can I manage special devices other than SNMP standard devices like switch and router?

To manage special line, network, dial, etc. you have to define your own enterprise MIB or you have to find out some standard MIBs if they are available. To write new enterprise MIB use our MIB Editor Tool. Once MIB is ready then you can use MIB Compiler for the code generation. During agent instrumentation you have to modify the generated code to retrieve data from your device.

Please refer to the Tutorials topic in each "Building the Agent" section of the help documentation to know more.

20. How does your toolkit help us rapidly develop, test and deploy Agents?

Our product provides more powerful and intuitive visual development and test tools for developing the Agent of your choice. Since our product has all types of tools required for agent development, the agent deployment will be much easier.

21. Can an agent have multiple clients (e.g., EMS + direct SNMP monitoring access)?

Yes, any number of Manager can access the Agents. There is no restriction as to how many managers can access an agent.

22. How do you run the Agent Toolkit using Visual Cafe?

Edit <Visual Cafe's Installation Directory>/bin/sc.ini and modify the CLASSPATH value as given below
CLASSPATH =<the>;%CLASSPATH%
Release Notes

Release Features: This section gives the details of the features of the current release of the product. It also lists the enhancements to the product starting with the Release 2.x of the product up to the current release.

Known Issues and Limitations: It lists the known issues and limitations of the product.

Bugs Fixed: Some of the bugs fixed after 5.0 release of Agent Toolkit are mentioned here.

Agent Toolkit Jars Description: This briefly describes the jars packaged with the Toolkit.

Package Structure: This section lists the various directories and files that are distributed with the AdventNet Agent Toolkit.

Third Party Software: This section lists the various third-party software packages that are bundled with this product.
Release Features

- Changes from Release 5.1.0 to 6.0.0
- Changes from Release 5.0 to 5.1.0
- Changes from Release 4.2 to 5.0
- Changes from Release 4.1 to 4.2
- Changes from the release 4.0 to 4.1
- Changes from the release 3.2.1 to 4.0
- Changes from the release 3.2 to 3.2.1
- Changes from the release 3.1.1 to 3.2
- Changes from the release 3.1 to 3.1.1
- Changes from the release 3.0 to 3.1
- Changes from the release 2.1 to 3.0

6.0.0 Release Features

SNMP Agent
- Developing and porting the agent in J2ME CDC Environment to address devices that requires less-memory and limited-resource requirement
- Introducing Rule Engine that defines users’ logic as rules to monitor the attributes of users' application/device and then execute some actions.
- Support for IPV6 that is a new type of addressing model for communicating between agent and manager. The implementation are followed in TrapForwardingTable to address the host type with IPv6 model.
- Introducing an Admin GUI management application for managing SNMPv3 tables such as USM and VACM tables.
- Enhanced Multi-varbind functionality such that the agent will not process each request one after another, but groups all the varbinds under a parent OID as a single request before calling the instrumentation layer.
- Supports CFB-AES-128 privacy protocol for encryption in SNMPv3.
- Compliant to the latest RFC (341X Series) Standards.
- Providing more flexibility in sending traps where it is possible to add the variable bindings to Trap PDU by just specifying the name of the object.
- Binding SNMP Trap Service to a particular IP Address.
- Trap Receiver Port configurable even without enabling Trap Filtering Table.
- Starting Agent as Service in Windows
- Support of SNMPv3 Inform

Multi-Protocol Agent

SSL (Secure Sockets Layer) support for RMI Adaptor.
5.1.0 Release Features

Multi-Protocol Agent

- Authentication mechanism implementation has been changed for RMI and CORBA. Now it is exposed as one of the Service MBean.
- JMX-SNMP Proxy handling has been made XML-driven which provides more flexibility.

SNMP Agent

- Disabling of StartUp Traps functionality added
- Support for SNMP V3 Request in Context Based Proxy is provided. AdventNet Agent can be configured as a subagent to LINUX Agent.

General

- Standard License is changed to E-mail based License for trial users.
- Different types of license policy for developing SNMP, TL1, and Multi-Protocol Agent to the registered users. For more information, refer Using License Manager.

5.0 Release Features

SNMP Agent

Traps

- Support for Reliable Traps. Manager can be made aware of the number of Traps generated by the Agent and its details, using this support.
- Traps generated by an Agent are Validated and sent across to the Managers. This restricts the generation of unwanted Traps to the Manager.
- Support for Informs available.
- Support for specifying the Version of Traps, the TimeOut value, the Retries count and the Community in the Forwarding Tables.
- Multiple varbind support while sending SNMP Traps in JMX.
- SnmpAdaptor supports notif other than AttributeChangeNotifications.

Proxy

- Heart Beat Mechanism support to poll the status of the subagent.
- LinkUp and Link Down Traps to intimate the status of the subagent to the Master Agent.
- Getter and Setter methods provided for linkup and linkdown OIDs for SubAgentPoller.
- To add the extra varbinds in the SubagentLinkUp and SubagentLinkDown Traps when HeartBeat mechanism is supported.
- Trap Filtering Table to filter the traps and send it to the Managers through the Master Agent.
- Facility for sending Traps through Windows NT Master Agent to the Manager.
- setVersion() is added in SnmpProxy to let users specify the SNMP version to be used while proxying requests from the Master Agent.
- Support for specifying the Version, TimeOut, Retries, Community details for proxying the Requests to the Subagent.

General

- Two new methods have been introduced, namely createAndAddEntry and createAndAddNewEntry methods for creating a row in a Table using api calls.
- SNMP Configuration Tables /API Tables exposed through all the Adaptors.
• **Agent Tree Performance** is improved.

• Facility is provided for **disabling Atomicity** Support.

• **Root OID Configuration** facilitates to configure the root oid before generation.

• **AGENT-SNMP-CONFIG-MIB** A new MIB which contains all the SNMP Configuration Tables like Proxy Table, Trap Table etc.,

• **Agent-Sample-MIB** can be used as a testing mib.

• Various **Logging** levels helps in tracing the problems arising during development. Debugging messages get generated based on the level chosen.

• **Commandline options** are available for directly specifying the path of the conf file and the dll while running the proxyreg.exe

• **SNMPv3 related table** data can be stored in XML file format, Ser file format and also in Runtime Memory.

• **Lo4j utility** implemented to send Traps for Log messages.

• **Failover support for API Table** on Agent crash is available.

**MIB Compiler**

• SNMP Agent Table or API Table configurations (like aclTable, Trap Forwarding Table, Proxy Table) made available in the UI.

• Default entries are also available in the API Tables.

• New Workspace and Project Handling options.

• Storage Models are also newly introduced in the UI for all API Tables and Ordinary Scalars and Tables.

• Settings related to V3 have been introduced.

**Multi-Protocol Agent**

• **Authorization**: View Based Authorization has been supported to restrict the access of users.

• **SSL**: HTTP Connector and HTML Adaptor now support communication using SSL (Secure Sockets Layer).

• **Enhanced Adaptor and Client framework**.

• **Authentication**: Authentication support has now been provided for RMI Adaptor, CORBA Adaptor, and HTTP Connector.

• **Table Monitor**: Support has been provided for monitoring Table attributes in addition to the existing support to monitor String, Counter, and Gauge attributes.

• **Logging**: Support has been provided to generate log messages at agent runtime. Specific logging levels can also be specified.

**JMX Compiler**

• Support for adding JMX Subagent entries for Cascader from compiler user interface.

• Support for adding Multiple SNMP Subagent entries for SNMP Proxy from the UI

• Management Applet support

• Support to configure Discovery service parameters

• Support for configuring Timer service parameters

• Support for monitoring String, Counter, Gauge and Table attributes through Monitor service

• Support to configure settings pertaining to SSL support for HTTP Connector and HTML Adaptor.
Getting Started

- Support to define Usernames for Authentication in HTTP Connector, RMI Adaptor, and CORBA Adaptor
- Support to configure parameters regarding User security, Channel security, Operation security, Command security and Resource security for the TL1 Adaptor.

MBean Browser
- Support to discover multiple agents.
- Support to directly connect to any agent that has been discovered using HTTP, RMI, or CORBA protocol.
- Heartbeat support.
- Option to register/unregister an MBean in the MBean server

TL1 Agent
- TL1 Agent Conforms to GR 815, 833 and TR 835 standards. Also Provisioning Messages from GR199 has been implemented
- Security: security feature in TL1 Agent with rich set of security options has been implemented as per the GR 815 & TR 835 standards.
- Asynchronous Message Processing: This enables the client to send many requests simultaneously without having to wait for the previous message to be processed.
- Delayed Activation Support which enables an input message to be stored in a message pending buffer at the NE for final execution at some later time
- Built in Commands support which can help you control the TL1 Agent with much ease.
- Built in Commands for Delayed Activation support which can help you activate, deactivate or retrieve Delay Activation input messages.
- Transport Provider Interface support which helps you plug in any transport protocol.
- Self Monitoring support which helps you to monitor the Agent details like the number of requests received, number of responses and autonomous messages sent etc.
- Craft Interface Support: The input commands can be typed on the console from where they are sent to the TL1 Agent for processing.
- A TCS file has been defined for all the commands in GR 833.

TL1 Compiler
- Support to Create, Modify and Configure the five security views namely, User security, Channel security, Operation security, Command security and Resource security for the TL1 Agent.

TL1 Message Builder
- Complete Modifier 2 Support in the TL1 Message Builder

Changes from Release 4.1 to 4.2

The AdventNet Agent Toolkit (Java Edition), release 4.2 has new tools included like TL1 Message Builder (for defining TL1 command set file) and TL1 Compiler (generates standalone TL1 agent for the given .tcs file). Apart from these new tools, the features included in 4.2 release, are listed below, under the following modules:

- Multi-Protocol Agent
- SNMP Agent
- TL1 Agent
- MibCompiler UI and Code Generation
- MibEditor
Getting Started

- TL1MessageBuilder
- TL1Browser

Multi-Protocol Agent
- JMX Compiler: An intuitive UI introduced for generating a Multi-Protocol Agent.
- It provides comprehensive solution using Model MBeans for efficiently handling tables.
- User Storage Model In Table Handling: Allows the user to have the storage of his own and he just needs to provide access to get it managed. JmxTableModelListener interface is added to serve this purpose.
- TL1 protocol is added in common instrumentation along with the existing SNMP, HTTP, RMI and CORBA adaptors.
- In Multi-Protocol Agent, HTML Adaptor has provision for plugging in a web-server of user's choice, instead of using the Jetty web-server by default.
- Discovery service support: The discovery service is an MBean, which enables to discover the AdventNet Multi-Protocol Agents in a network.
- Multi-protocol access to existing SNMP Agent.
- Heart Beat Support in JMX - Connector clients can use this feature to check whether their connectors are alive or not in the Multi-Protocol Agent.

SNMP Agent
- Support for Disabling Rollback Method.
- Heart Beat Mechanism support to poll the status of the subagent.
- LinkUp and Link Down Traps to intimate the status of the subagent to the Master Agent.
- Trap Filtering Table to filter the traps and send it to the Managers through the Master Agent.
- Subid list and value Mapping Feature to expose the subids at PDU Request Handler level.
- Logging Option for the benefit of tracing any problem arising during development.
- Root OID Configuration facilitating the configuration of root oid before generation.
- Traps sent to Managers can be Validated and sent across. This would avoid unwanted traps reaching the manager.
- Support for Reliable Traps.
- CVM Support.
- Facility for sending Traps through Windows NT Master Agent to the Manager.
- Commandline options for directly specifying the path of the conf file and the dll while running the proxyreg.exe
- The newly created methods in Table Handling namely createAndAddEntry, createAndAddNewEntry and createInstance methods helps in creating a row to the table using api calls.
- Agent Tree Performance has been improved.
- Agent-SNMP-Config-MIB has been introduced with all the configuraton tables under it. Agent-Sample-MIB can be used as a testing mib.

TL1 Agent
- TL1 agent conforms to Bellcore GR 831 standard.
- Support to generate autonomous messages.
- Support for authentication with Log in and Log out services.
- Multiple port support.
MibCompiler UI and Code generation
- The MibCompiler tool provides a "Project Workspace" for the users to create multiple agents for different projects.
- Code generation settings enhancement including configuring SNMP proxy information.
- Support in code generation for XML persistency for tables is provided as XMLToVector option in UI.
- Enhanced code merging to facilitate easy migration between the releases.
- Support for generating .tcs (TL1 command set to be loaded in TL1 Browser) and .dat (data file, which has default values for the generated tcs file, to be loaded in the TL1 Browser) file during JMX code generation when TL1 adaptor is selected.
- Start and stop agent option in Mibcompiler to run the agent from the Compiler UI

MibEditor
- Support for defining MacAddress syntax.
- "Find" based on OID, possible in both MibEditor & MibCompiler.
- Support for defining "networkAddress" SYNTAX as index for tables.

TL1MessageBuilder
- Provides easy-to-use wizard interface to create and modify TCS files (TL1 command set files).
- Support to define the TL1 command code, consisting of a verb followed by up to two other optional modifiers.
- Support to define the Access Identification parameters for the TL1 command code.
- Support to define the Message Payload block parameters for the TL1 command code.
- Support to define the Response parameters for the TL1 command code.

TL1 Browser
- Support to load either the TCS file or Text file in the TL1 tree.
- Support to load more than one file at a time.
- Provides text fields to display all the details of the input messages like command code, TID, AID, Correlation Tag, General Block and Message Payload block.
- Provides text area to display response messages, autonomous messages and log messages.

General
- Interpreted approach of agent development has been deprecated from 4.2 release. It is advisable to use the compiled approach for developing an agent.
- AgentSimulator tool is no longer bundled with Agent Toolkit (Java Edition). This tool is bundled with another product, AdventNet Simulation Toolkit. You can download the latest version of this product from www.adventnet.com.

Changes from the release 4.0 to 4.1

Agent Simulator
- User-Interface completely modified
- The user-interface has been completely modified providing the user with a more intuitive, easy-to-use and user-friendly interface.

Saving a Configuration made Optional
- In the previous release of the Agent Simulator, saving a configuration was a necessary step before starting the agent, as the agent read the simulated values from the configuration files.
Thus one had to wait for the process of saving to be get completed before starting the agent. In this release of the Agent Simulator, saving a configuration has been made optional. The agent can be started as soon as the MIB gets loaded or after the user has finished specifying values for the nodes. The configuration can be saved for future references.

Dynamic Updation of data

- In the previous release of the Agent Simulator, everytime a value for a node was modified or an instance added or deleted to a table, the configuration had to be re-saved and the agent restarted for the changes to get reflected. In this release the changes get reflected dynamically. There is no need for either the configuration to be resaved or the agent to be restarted.

Enhanced Trap Generation Capability

- **Traps based on Threshold and Timer:** In the previous release, traps could be generated only for every GET / SET done on an OID. In this release more powerful trap generation capability has been added. Traps can be generated based on a user specified threshold value. Sending of traps can also be delayed by configuring a timer. Purely timer based traps can also be generated.

- **Trap Generation from the user-interface:** In the previous release there was no mechanism by which the user could generate traps directly from the UI. In this release such a mechanism has been provided giving the user a finer control for generating traps. There is no need for the agent to be started for sending traps by this method.

- **V3 Trap is supported:** Generic and Specific Type for v1 trap supported.

- **RowStatus Support:** The SNMPv2-TC RowStatus, " which is used to manage the creation and deletion of conceptual rows ", is fully supported.

- **Command Line Simulator:** The simulated agents can be started from command line. This helps in running multiple agents listening in different ports.

Runtime agent

Recorder and PlayBack bean in com.adventnet.snmp.snmp2.agent package is removed in this 4.1 release. Since these beans function is used and enhanced in Agent Simulator tool, we would recommend the existing users of these beans to use the Agent Simulator.

ClassCommand in AgentClause is removed from the AgentClause definiton in Integrated Approach. Existing users who uses CLASS-COMMAND in Interpreted agent can generate code for their MIB using MibCompiler and copy the stubs written for the CLASS-COMMAND to the generated instrument/entry files.

Tools

AgentIDE tool now does not have the MibEditor and MibCompiler tab. Thus it plays the role of pure Java Bean Builder. Hence it is renamed as Java Bean Builder.

Other Features Supported in 4.1

TL1 agent developer who uses the TL1 adaptor on top of the JMX server, for testing the developed agent, is provided with a manager side tool called TL1Browser, by the Agent Toolkit. For more details refer to the TL1 Agents section under Developer Guide.

AdventNet Agent Toolkit supports interface mode of code generation. By this, user need not edit any generated Java files from MibCompiler. Thus user need not worry on migrating to releases (or) upgrading the MIB knowledge of SNMP agent. This feature is supported in JMX code generation too. It now supports Monitor and Timer service in JMX. HTTP Connector and client are added in this new release for JMX support. The Snmp Agent now supports instrumentation of SNMPv3 notification table, acl table for SNMPv2 and implements groups for SNMPv2 compliance. The Snmp Agent now supports SNMP Transport Provider Framework whereby the user can plug-in their own transport protocol.
SNMP Transport Provider Framework

A new protocol-independent transport provider framework has been provided for SNMP communication, whereby the user can plug-in their own transport protocol. This essentially means that the user can use AdventNet SNMP over any transport protocol (like UDP/IP, TCP/IP, Serial link and so on). This mechanism can also be used by users requiring more security to run SNMP over SSL. UDP/IP, is the default protocol implementation with a reference implementation for TCP/IP, is also provided. The following interfaces are newly added for the SnmpTransport provider framework:

- ProtocolOptions
- SnmpTransportProvider

Users wishing to use their own protocols below SNMP for transport communication should implement this interface.

A new class SnmpTransportPacket has been included which consists of the details of the parameters that are needed for the transport communication and the details of the message data that is to be sent or received. The following are the new classes which are TCP reference implementations that have been added in the com.adventnet.snmp.snmp2.agent package:

- AgentTcpProtocolOptionsImpl - This is a TCP reference implementation of the ProtocolOptions interface
- AgentTcpTransportImpl - This is a TCP reference implementation of the SnmpTransportProvider interface

Changes from the release 3.2.1 to 4.0

- **API Changes:** The mibs package uses new MibParser which gives better performance. The following are the changes in MIB related operations due to new MibParser.
  
  - All the modules defined in the import field should be present. Earlier even if imports like, mib-2 from RFC1213-MIB failed it will load the MIB after printing a warning message. Now it is mandatory that all the modules defined in the imports are present.
    
    - Oid definition like the following will not work.
    - newNode OBJECT IDENTIFIER ::= { ModuleName.nodeName 1}
    - newNode OBJECT IDENTIFIER ::= { 1 }
    - newNode OBJECT IDENTIFIER ::= { 1 3 6 ........}
    - Module name should not start with a lower case letter or number
    - Enumeration label should not start with a number.
    - Since multiple roots are shown in the MibTree now, get and getnext operation from the MIB browser after selecting the module name will fail now. You have to select the mib node.

- Three new methods will be generated for handler file in the code generation for embedded agent. They are,
  
  - <ModuleName>ProcessRequests() - It will have a vector that contains requested OIDs in the received request and PDU type. So the user can do validation on the received OID before doing processing on them in the <ModuleName>ProcessReadReq() and <ModuleName>ProcessWriteReq() methods.
  - Update<ModuleName>() - It will be called after SET success for all received OIDs to update exact data structure. It is used in the case of no storage for snmp table.
  - free<TableName>Vector() - It is not a API method. It is used to free the table. It is only for snmp table.
  - Support for SNMPv3 USM and VACM remote configuration. ie. RFC2574(USM-MIB) and RFC2575(VACM-MIB) are implemented.
Changes from the release 3.2 to 3.2.1

MibEditor
- Support for definition of AUGMENTS entry and EXTERNAL INDEX in the add node wizard.
- Support for definition of complex MIB macros like MODULE-COMPLIANCE and AGENT-CAPABILITIES.
- Removed the Global MibView Vs Module Level View of the loaded modules.

MibCompiler
- Support for Java Management Extension (JMX) - Agent and Instrumentation level.

Changes from the release 3.1.1 to 3.2

MibCompiler
- IMPLIED index handling in code generation.
- DateAndTime (SNMPv2-TC) TC is handled in code generation.
- Code merging for RequestHandlers and Main file.
- Common Base class for xxxEntry.
- Debug messages in xxxTableRequestHandlers.
- CheckBox for FileToVector generation, Trap code generation.
- Common base class for xxxTableRequestHandlers.
- Option to initialize variables generated to null.
- Writing to fileToVector is done more efficiently.

Others
- ccitt path registration handling.
- Multiple indices in Database table is handled.(SnmpDBAdaptor).
- Supports Minimal featured Database like Timesten, CloudScape etc.
- Provided a dll apadll.dll (only for Windows NT systems) which registers as a sub-agent to Windows NT master agent and acts as a proxy-forwarder i.e., forwards the request to the specified SNMP agent running in a different port.

Changes from the release 3.1 to 3.1.1
- Running the agent in asynchronous mode.
- Two aeadll.dll are provided (this is the dll which starts the AdventNet's SnmpAgent as a sub-agent in Windows). One dll in the <Agent Toolkit>/JavaEdition/dll directory is jdk1.1.x compliant and the other in <Agent Toolkit>/JavaEdition/dll/jdk1.2_dll is jdk1.2.x compliant.

Changes from the release 3.0 to 3.1

MibEditor
- The AGENT-CLAUSE statements can now be saved in a separate file, keeping the MIB file unchanged.
- The MIB tree supports Global MibView Vs Module Level View of the loaded modules.
- Finds a node in the tree through Find Dialog.
- Support for Copy/Paste/Rename of leaf nodes.
- Reloading of loaded modules.
MibCompiler

- Support for Atomic SET (RollBack).
- Support for Snmp group counters.
- Support for External index in code generation.
- Generating code for converting Files to Vector (Helpful in replacing FILE-COMMAND).
- Multiple index support in code generation.
- Multiple MIB support while code generation.
- Generating code for traps.
- Generating validation code inside SET.

Others

- SnmpV3 support.
- Record and Playback support (For Network Simulation).
- Multiple agent on single JVM aeadll.dll supports traps in the usual NT method for an extension dll.
- Database access primitives for making Databases accessible via SNMP.

Changes from the release 2.1 to 3.0

MibEditor

- MIB definitions wizard provides a single user interface for defining constructs.
- Support for NOTIFICATION-GROUP and OBJECT-GROUP has been added.
- MODULE-COMPLIANCE and AGENT-CAPABILITIES are not yet supported for editing. (A MIB file with these two constructs can still be loaded in the Agent Toolkit, the support is lacking only in the wizard interface.)
- "Save as" SMIv1 and SMIv2 have been provided for conversion between V1 to V2 and vice versa.
- Support for publishing MiBs without AGENT-CLAUSE hooks have been provided.

API Changes

- The package Snmp2.* is no longer supported.
- Changed interfaces for CLASS-COMMAND. OID values now take in int arrays (instead of long arrays).
- Removed MIB dependencies for SnmpAgent and com.adventnet.snmp.snmp2.agent.*.
- Packaging of smaller minimal agents are possible with the help of compiled approach.
- The interface com.adventnet.snmp.mibs.agent.InitSnmpAgentData doesn't pass in SnmpAgent reference. SnmpAgent reference was required in earlier versions for TRAP support which can be obtained through use of SnmpTrapService and SnmpMibTrapService now.
- Made TrapForwardingTable an interface. API users can now provide their own implementations of TrapForwardingTable.
- Support for dynamically registering SNMP sub agents.
- Support for Access Control Lists to authenticate read and read-write access to managers.
Known Issues and Limitations

The Known Issues topic lists the known issues with the current release of the product. We, at AdventNet, take all efforts to keep this page updated to save your valuable time for you. So, please drop us a word about any problems you encounter with this product to agent-support@adventnet.com. Also check out our website for the Latest Known Issues page.

The Limitations topic describes the known limitations of the product. If you find any limitation with our product during your development, please feel free to write to us at agent-support@adventnet.com. We will try to meet your requirements and improve the scope of our product.

Known Issues

General
- Even if the logging level is chosen as 0, the log file will be created.

Multi-Protocol Agent
- Apart from "Object Reference" the following references are not supported by the MR type (managed resources), in Model MBeans.
  - Handle
  - IOR
  - EJBHandle
  - RMIReference

JMX Compiler
- When an Agent started from the JMX Compiler is stopped from the compiler user interface, some of the ports may not be cleaned. (This issue is specific to Linux OS.)
- In Linux OS, with JDK1.2.2, the HTML adaptor will not be cleaned up. In JDK1.3 the TL1 Adaptor Server socket session will not be closed and the HTTP port will not be cleaned up.
- The tools that are present in the Components bar/Tools Menu cannot be invoked from the Compiler, if you have started the JMX Compiler from the launcher

SNMP Agent
- In External Index, if a row in the Base Table is deleted, the corresponding row in the Derived Table will not get deleted.
- Embedded JVM support (GCJ, Newmonics,VAME) is compatible only with release version 4.2. It is not upgraded to the current version.
- While defining a new TC, if Octet String is used as base type it will not be handled as a byte array in the generated code.
- The SNMP V3 administration tool currently does not support AES privacy protocol.
- Whenever SNMP GET or GET-NEXT request is processed for a particular attribute of a scalar group, in the generated code, the getAttributes() method of InstrumentHandlerInterface class is called which retrieves the data for all the attributes in the scalar group.

MIB Editor
- Modification: While modifying a MIB in the MIB Editor the following construct is not allowed for modification: TEXTUAL CONVENTIONS
- SMI version differences: Syntax list contains syntaxes corresponding to SMI version, for defining scalars and Tables. In the syntax list, basic syntaxes and well known TCs are alone differentiated based on the SMI version. You have to take care of SMI version for all other imported TCs.
- DefaultValue: The validation is done for all basic syntaxes as per the format. If you use TCs, then the format of the parent's default value will be considered for validation. In case of
OBJECT IDENTIFIER, number format is only accepted as default value. You can't give an identifier as a default value for object which has syntax as an OBJECT IDENTIFIER.

- MIB Editor doesn't allow to create more than one MIBMODULE in a single mib file. But you can load and modify the MIB files which contains more than one MIBMODULE.
- Hyphens in node names does not create any problem during conversion of a v1 mib to a v2 mib. When the SNMPv1 mib having node names with hyphens is converted to an SNMPv2 mib the hyphens are taken as such. It is not necessary to make any modification in the node name as specified in the standards.
- The "OPTIONAL" status in SNMPv1 MIB becomes CURRENT when the V1 MIB is converted to a V2 MIB.
- The Units sub-clause information required for MIBs is not supported by the MIB Editor of AdventNet Agent Toolkit.

MIB Compiler
- The tools that are present in the Components bar/Tools Menu cannot be invoked from the Compiler, if you have started the Compiler from the launcher.

TL1 Agent
- Known issues related to JDK 1.3 in RedHatLinux: When we do a session.close() in the application, the socket is not closed properly and the session is still alive. More Information on this issue can be seen in the Java Developer Connection with the Bug ID: 4344135. http://developer.java.sun.com/developer/bugParade/bugs/4344135.html
- In TL1 Compiler, the tools that are present in the Components bar/Tools Menu cannot be invoked from the Compiler, if you have started the Compiler from the launcher.

Limitations

SNMP Agent
- Rollback support with set request in RowStatus column between Master and SubAgent is not supported.
- Support for Multiple Agents in single JVM is absent for Snmp Transport Provider Framework default implementation.
Bugs Fixed

- Bugs Fixed in 6.0.0 Release
- Bugs Fixed in 5.1.0 Release
  - SNMP Agent
  - Multi-Protocol Agent
  - TL1 Agent
  - Tools

Bugs Fixed in 6.0.0 Release

Some of the important bugs fixed after the 5.1.0 release of AdventNet Agent Toolkit are listed below:

- The performance of Mib Compiler has been improved. The amount of time taken to generate agent source if target directory is in the remote network drive has been reduced greatly.
- DateandTime syntax allows octet string of length 8 and 11 bytes. Thus, the methods dateString() and dateStringToByteArray() in CommonUtils.java have been modified to handle both 8 and 11 bytes.
- Provision for redirecting logging has been incorporated.
- In SNMP DB Adaptor, ConnectionManager has been introduced to reuse the database connections.
- While terminating the SNMP agent, all the threads started by it and other allocated resources will be cleared perfectly.

Bugs Fixed in 5.1.0 Release

Some of the bugs that have been fixed after the 5.0 release of AdventNet Agent Toolkit are listed under different modules:

SNMP Agent

- The instance value with the Scalar VarBind OID’s sent along with the Agent Traps are appended
- The TimeStamp value in the Traps sent out from the Agent is getting updated
- SNMP Request processing problem when a Table is defined within a Scalar Group is fixed.
- SubAgentPoller related Threads closure problem is fixed.
- Duplicate VarBinds while forwarding SubAgent Traps were removed
- SNMP Proxy failure, if the ProxyOID is the last registered OID in the Master Agent is solved.
- Request Processing issues in Proxy architecture were fixed
- Code generation Related Issues in MIB Compiler were fixed
- Error handling in Standalone as well as Proxy Agents is improvised
- Issue with NoAccess community error handling has been fixed
- SNMP Examples were improvised.
**Multi-Protocol Agent**

- `setHostAdress(String IpAddress)` is added in DiscoveryService.
- External Index Handling is done in JMX Snmp Proxy.
- Problem in updating the `snmpEnableAuthenTraps` variable through protocol Adaptors other than SNMP Adaptor is resolved.
- UserStorageModel code generation problem in JMX Compiler is fixed.
- Improper removal of Notification ID in Timer Service is fixed.

**TL1 Agent**

- Problem while performing a tl1 request without "<" symbol is solved.
- UserRequestHandler interface is added to extend TL1Request and TL1Response processing.

**Tools**

- Code merging problem when user codes are added above the package declaration of the generated source is fixed.
- Table index and column conversion based enhancements and issue fixes in MIB Editor is done.
- Alternative workaround while loading workspace where the project location specific in it doesn't exist is being fixed.
- KeyListener based issues occurring with JDK1.4 is fixed.
- Problem while loading a project file from a directory other than default directory is solved.
- Special characters encoding done while writing into Framework based XML files.
- Problem with receiving String Monitor, Gauge Monitor and Counter Monitor Notifications by the RMI Client is solved.
## Package Structure

The AdventNet Agent Toolkit package is installed under `<Agent Toolkit Home>` directory from wherever the files are extracted. The directory structure and the files under `<Agent Toolkit Home>` directory are as follows:

<table>
<thead>
<tr>
<th>Directory</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>bin</td>
<td>Contains shell scripts/batch files to run each tool as a separate application.</td>
</tr>
<tr>
<td></td>
<td>- HtmlBrowser.bat/sh - used to invoke WebBrowser</td>
</tr>
<tr>
<td></td>
<td>- JMXAgentSourceGen.bat/sh - command line tool which is used to generate Multi-protocol stub files</td>
</tr>
<tr>
<td></td>
<td>- setenv.bat/sh - It defines environmental variables which are used in other scripts and batch files.</td>
</tr>
<tr>
<td></td>
<td>- SnmpAgentSourceGen.bat/sh - command line tool which is used to generate Standalone SNMP stub files</td>
</tr>
<tr>
<td></td>
<td>- SnmpMigrationTool.bat/sh - Tool to convert 4.2 generated configuration files to 5.1.0 generated files.</td>
</tr>
<tr>
<td></td>
<td>- SnmpV3AdminTool.bat/sh - Tool to configure USM tables and VACM tables for SNMPv3 agent.</td>
</tr>
<tr>
<td></td>
<td>- TL1AgentSourceGen.bat/sh - command line tool which is used to generate Standalone SNMP stub files</td>
</tr>
<tr>
<td></td>
<td>- UpdateManager.bat/sh - used to install service pack.</td>
</tr>
<tr>
<td></td>
<td>- JMXCompiler.bat/sh - used for running the JMX Compiler application.</td>
</tr>
<tr>
<td></td>
<td>- MibBrowser.bat/MibBrowser.sh - to run the MIB Browser.</td>
</tr>
<tr>
<td></td>
<td>- MibCompiler.bat/MibCompiler.sh - to run MIB Compiler.</td>
</tr>
<tr>
<td></td>
<td>- MibEditor.bat/MibEditor.sh - to run MIB Editor.</td>
</tr>
<tr>
<td></td>
<td>- MBeanBrowser.bat/MBeanBrowser.sh - to run MBean Browser.</td>
</tr>
<tr>
<td></td>
<td>- TL1Browser.bat/TL1Browser.sh - to run TL1 Browser</td>
</tr>
<tr>
<td></td>
<td>- TL1Compiler.bat/TL1Compiler.sh - to run the TL1Compiler</td>
</tr>
<tr>
<td></td>
<td>- TL1MessageBuilder.bat/TL1MessageBuilder.sh - to run the TL1 Editor</td>
</tr>
<tr>
<td></td>
<td>- v3apps.bat/v3apps.sh - to run the v3 configuration tool.</td>
</tr>
<tr>
<td></td>
<td>- JmxXMLTranslator.bat/.sh - Batch file used for translating the XML files upgraded from 4.1 to 4.2 release.</td>
</tr>
<tr>
<td>conf</td>
<td>apiTablesxml - contains the Model MBean descriptor files for configuration tables supported by Multi-protocol agents.</td>
</tr>
<tr>
<td></td>
<td>settings - property files for compiler tools</td>
</tr>
<tr>
<td></td>
<td>defaultV3Settings - snmpv3.ser and defaultsnmpv3.ser has the default configuration for SNMPv3 USM, VACM, Community Authentication and Notification Filtering.</td>
</tr>
<tr>
<td></td>
<td>appenderConfig - sample configuration file for configuring SNMP Trap Appender.</td>
</tr>
<tr>
<td>connector</td>
<td>Contains all files related to HTTP Connector</td>
</tr>
<tr>
<td>dll</td>
<td>Contains dll files which are used in the packaging of sub-agent for windows.</td>
</tr>
<tr>
<td></td>
<td>proxy - contains config file for SNMP subagent registration to Windows native agent.</td>
</tr>
<tr>
<td>Directory</td>
<td>Files</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>directory</td>
<td>contains various example files, organized as shown below.</td>
</tr>
<tr>
<td>examples</td>
<td>- jmx: contains examples that illustrate various features supported by multi-protocol agent.</td>
</tr>
<tr>
<td></td>
<td>- snmp: contains examples that use snmp api and various features of snmp agent.</td>
</tr>
<tr>
<td></td>
<td>- tl1: contains examples that use tl1 api and features of tl1 agent. it also contains the tl1 tutorial.</td>
</tr>
<tr>
<td>etc</td>
<td>contains all property files for html adaptor and http connector.</td>
</tr>
<tr>
<td>help</td>
<td>contains the complete help files in html format. folders being.</td>
</tr>
<tr>
<td></td>
<td>- images</td>
</tr>
<tr>
<td></td>
<td>- gettingstarted</td>
</tr>
<tr>
<td></td>
<td>- mp_agent - this includes building multi-protocol agent related documentation as well as its javadocs.</td>
</tr>
<tr>
<td></td>
<td>- snmp_agent - this includes building snmp agent related documentation as well as its javadocs.</td>
</tr>
<tr>
<td></td>
<td>- tl1_agent - this includes building tl1agent related documentation as well as its javadocs.</td>
</tr>
<tr>
<td>images</td>
<td>contains the images used by the adventnet agent toolkit.</td>
</tr>
<tr>
<td>jars</td>
<td>contains the jar files used by adventnet agent toolkit and they are listed in agent toolkit jars description section.</td>
</tr>
<tr>
<td>lib</td>
<td>contains libraries for licensing.</td>
</tr>
<tr>
<td>jmxprojects</td>
<td>multi-protocol agent projects created using jmx compiler will be under this folder. this folder is created only when a project is generated using jmx compiler.</td>
</tr>
<tr>
<td>logs</td>
<td>contains the log files - *.err and *.out - which stores the error and output messages</td>
</tr>
<tr>
<td>mibs</td>
<td>contains standard mib files, configuration mibs and the sample mibs supplied by adventnet. any mib file created using the mib editor gets stored in this mib folder by default.</td>
</tr>
<tr>
<td>reference</td>
<td>- jmx - a reference implementation has been used for explaining multi-protocol agent development and the features supported by multi-protocol agents. the docs folder contains the help files for the reference implementation.</td>
</tr>
<tr>
<td></td>
<td>- snmp - this reference implementation directory contains a shopping cart application for explaining the features supported by snmp agents. the docs folder has the help files required for understanding the shopping cart application.</td>
</tr>
<tr>
<td></td>
<td>- shoppingcart - sources and binaries related to shopping cart application.</td>
</tr>
<tr>
<td>snmpprojects</td>
<td>snmp agent projects created using mib compiler will be under this folder. this folder is created only when a project is generated using mib compiler.</td>
</tr>
<tr>
<td>subagents</td>
<td>- ntutils - contains utility files for nt subagents.</td>
</tr>
<tr>
<td></td>
<td>- solarisutils - contains utility files for making solaris subagents.</td>
</tr>
<tr>
<td>tcs</td>
<td>contains all the tcs files and the data set files. any tcs file created using the tl1messagewriter gets stored in this tcs folder by default.</td>
</tr>
<tr>
<td>tl1projects</td>
<td>tl1 agent projects created using tl1 compiler will be under this folder. this folder is created only when a project is generated using tl1 compiler.</td>
</tr>
<tr>
<td>tomcat</td>
<td>contains all files related to tomcat webserver.</td>
</tr>
<tr>
<td>Directory</td>
<td>Files</td>
</tr>
<tr>
<td>----------------------------</td>
<td>---------------------------------------------------------</td>
</tr>
<tr>
<td>Agent Toolkit.sh/Agent Toolkit.bat</td>
<td>Shell script/Batch file for running Agent Toolkit.</td>
</tr>
<tr>
<td>COPYRIGHT</td>
<td>Copyright file.</td>
</tr>
<tr>
<td>LICENSE AGREEMENT</td>
<td>License Agreement file.</td>
</tr>
<tr>
<td>README.html</td>
<td>README file for the AdventNet Agent Toolkit.</td>
</tr>
</tbody>
</table>
# Agent Toolkit Jars Description

This document describes the jars present in AdventNet Agent Toolkit Java Edition 6.0.0 product.

<table>
<thead>
<tr>
<th>Jar Name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdventNetJmxAgent.jar</td>
<td>This jar contains the Agent Toolkit Services, Client API, and classes for the Multi-Protocol agent.</td>
</tr>
<tr>
<td>AdventNetAgentWCF.jar</td>
<td>This jar is used for HTML adaptor</td>
</tr>
<tr>
<td>AdventNetAgentRuntimeUtilities.jar</td>
<td>This jar has utilities packages used by Multi-Protocol, SNMP, and TL1 Agent.</td>
</tr>
<tr>
<td>AdventNetCCLUtils.jar</td>
<td>This jar has classes related to MIB Browser</td>
</tr>
<tr>
<td>AdventNetCCLXMLParser.jar</td>
<td>This jar has classes related to MIB Browser</td>
</tr>
<tr>
<td>AdventNetJMacs.jar</td>
<td>This has the classes for JMac Editor used to edit java files in the various tools bundled with Agent Toolkit.</td>
</tr>
<tr>
<td>AdventNetLauncher.jar</td>
<td>It contains Agent Toolkit Launcher related classes</td>
</tr>
<tr>
<td>AdventNetNPrevalent.jar</td>
<td>License related information is provided in this jar</td>
</tr>
<tr>
<td>AdventNetLicense.jar</td>
<td>This also provides License related information</td>
</tr>
<tr>
<td>AdventNetLogging.jar</td>
<td>This jar has the AdventNet Logging Framework.</td>
</tr>
<tr>
<td>AdventNetSnmp.jar</td>
<td>This contains the low_level_snmp_api classes.</td>
</tr>
<tr>
<td>AdventNetSnmpAgent.jar</td>
<td>This jar contains the low_level_snmpagent_api classes.</td>
</tr>
<tr>
<td>AdventNetSnmpAgent_Gcj.jar</td>
<td>This jar contains the SNMP agent classes for GCJ.</td>
</tr>
<tr>
<td>AdventNetSnmpAgent_NEWMONICS.jar</td>
<td>This jar contains the SNMP agent classes for Newmonics.</td>
</tr>
<tr>
<td>AdventNetUpdateManagerInstaller.jar</td>
<td>This jar contains the API classes required for the Update Manager tool.</td>
</tr>
<tr>
<td>AdventNetTL1.jar</td>
<td>This jar contains the API classes required for TL1.</td>
</tr>
<tr>
<td>AdventNetTL1Agent.jar</td>
<td>This jar contains the low_level_tl1agent_api classes.</td>
</tr>
<tr>
<td>AdventNetJMXAgentTools.jar</td>
<td>This jar has MBean Browser, JMX Compiler and other multi-protocol agent related packages</td>
</tr>
<tr>
<td>AdventNetSNMPAgentTools.jar</td>
<td>This jar has MIB Editor, MIB Compiler and other SNMP related packages.</td>
</tr>
<tr>
<td>AdventNetJ2meSnmpAgent.jar</td>
<td>This jar contains the classes required for J2ME SNMP agent.</td>
</tr>
<tr>
<td>AdventNetRuleEngine.jar</td>
<td>This jar contains the API classes required for Rule Engine.</td>
</tr>
<tr>
<td>AdventNetSNMPv3AdminTool.jar</td>
<td>This jar contains the API classes required for the SNMP V3Admin tool.</td>
</tr>
<tr>
<td>AdventNetTL1AgentTools.jar</td>
<td>This jar has TL1 Message Builder, TL1Compiler and other TL1 related packages.</td>
</tr>
<tr>
<td>AdventNetUI.jar</td>
<td>It contains UI related packages</td>
</tr>
<tr>
<td>AdventNetAgentUIUtilities.jar</td>
<td>This jar has UI utilities for all tools</td>
</tr>
<tr>
<td>AdventNetAgentUtilities.jar</td>
<td>This jar has code merging, doc generation and other non-runtime, non-UI utilities</td>
</tr>
<tr>
<td>xmojo.jar</td>
<td>This jar has UI utilities for all Tools</td>
</tr>
<tr>
<td>crimson.jar</td>
<td>This jar contains JMX Implementation classes.</td>
</tr>
<tr>
<td>log4j-1.2.8.jar</td>
<td>It contains log4j utility classes required for third party tools.</td>
</tr>
<tr>
<td>MibBrowser.jar</td>
<td>MIB Browser tool related classes are provided.</td>
</tr>
<tr>
<td>Jar Name</td>
<td>Contents</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>servlet.jar</td>
<td>It contains packages for Tomcat Webserver</td>
</tr>
<tr>
<td>snmpTrapAppender_1_2_91.jar</td>
<td>This jar contains SNMP Trap Appender utility required for third party tools</td>
</tr>
<tr>
<td>webserver.jar</td>
<td>It contains packages for Tomcat Webserver</td>
</tr>
<tr>
<td>jasper.jar</td>
<td>It contains packages for Tomcat Webserver</td>
</tr>
<tr>
<td>jaxp.jar</td>
<td>This jar contains API classes required for third party tools.</td>
</tr>
<tr>
<td>xalan.jar</td>
<td>This jar contains API classes required for third party tools.</td>
</tr>
<tr>
<td>BuilderSwing.jar</td>
<td>This jar contains Swing beans like JDBC Adapter.</td>
</tr>
<tr>
<td>ApiUtils.jar</td>
<td>This jar has UI utilities for all Tools</td>
</tr>
</tbody>
</table>
Third Party Software

In addition to AdventNet's other products, third party softwares are also bundled with Agent Toolkit Java Edition to provide additional enhancement to the product. The following table lists the packages bundled with AdventNet Agent Toolkit Java Edition. Please see the Copyright file for details about the Copyright of the third party software and the respective License Agreement.

### Third Party Packages bundled with AdventNet Agent Toolkit Java Edition

<table>
<thead>
<tr>
<th>Package / Jar Name</th>
<th>Version</th>
<th>License Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tomcat</td>
<td>3.2.4</td>
<td>Covered under the Apache Software License Version 1.1</td>
</tr>
<tr>
<td>xmojo.jar</td>
<td>5.0</td>
<td>XMOJO Project, covered under Lesser General Public License</td>
</tr>
<tr>
<td>XML™ (jaxp.jar, crimson.jar, xalan.jar)</td>
<td>1.1</td>
<td>Covered under the Sun Microsystems, Inc Binary Code License Agreement and the Apache Software License Version 1.1</td>
</tr>
<tr>
<td>swingall.jar</td>
<td>1.1</td>
<td>Covered under the Sun Microsystems, Inc Binary Code License Agreement</td>
</tr>
<tr>
<td>JimiProClasses.zip (JIMI Software Development Kit)</td>
<td>2.0</td>
<td>Covered under the Sun Microsystems, Inc Binary Code License Agreement</td>
</tr>
<tr>
<td>log4j-1.2.8.jar</td>
<td>1.2.8</td>
<td>Covered under the Apache Software License Version 1.1</td>
</tr>
<tr>
<td>snmpTrapAppender_1_2_91.jar</td>
<td>1.2.91</td>
<td>Covered under the Apache Software License Version 1.1</td>
</tr>
<tr>
<td>JavaBeans Activation Framework</td>
<td>1.0.2</td>
<td>Covered under Sun Microsystems, Inc Binary Code License Agreement</td>
</tr>
<tr>
<td>JavaMail API</td>
<td>1.3.1</td>
<td>Covered under Sun Microsystems, Inc Binary Code License Agreement</td>
</tr>
</tbody>
</table>
Overview

The Simple Network Management Protocol (SNMP) is by far, the dominant protocol in Network Management. This Protocol (SNMP) was designed to meet network management needs with ease.

Please traverse through Building SNMP Agents guide to know more about building the SNMP Agent. AdventNet Agent Toolkit Java Edition helps to provide both Standalone SNMP Agent as well as Multi-Protocol agent that support SNMP. This guide provides a complete reference to build Standalone SNMP Agent. It also includes its architecture, features supported and its implementation.
Building Multi-Protocol Agents

Overview

The Agent Toolkit Java Edition provides intuitive development tools for building Multi-protocol agents. The developed agents provide multi-protocol access to common management information, including SNMP, RMI, HTTP, CORBA, and TL1. You just have to instrument once to access with any protocol.

The Multi-Protocol architecture is applicable to network management, remote system maintenance, application provisioning, and the new management needs of the service-based network.

To know more details about the building the agent, refer to Building Multi-Protocol Agents guide that highlights the architecture of the Multi-Protocol Agent explaining the key components, features of the product, and the development process for management applications.
Building TL1 Agents

Overview

The TL1 protocol is the backbone of network management in the telecommunications domain. Although TL1 is simple and easy to use, developing complex application becomes very difficult. The AdventNet Agent Toolkit Java Edition provides intuitive and innovative tools for rapid development of robust TL1 agents with much ease.

The Agent Toolkit uses the concept of XML-driven TCS files (TL1 Command Set file), which define the management information for the TL1 Agent. The TCS file is the equivalent of the MIB in an SNMP Agent.

Building TL1 Agents guide gives more detailed view on the steps involved in building the TL1 agent. It also explains TL1 architecture, the features supported and its implementations.