

IPvX

Table of contents

Welcome to IPvX	3
License Information	3
Introduction	4
About IPvX	4
System Requirements	4
Getting Help	4
User Interface	4
Menus	5
File Menu	5
IP Menu	6
Help Menu	6
Keypad	7
IPv4 Keypad	7
IPv6 Keypad	8
Value Fields and Labels	9
IPv4 Values and Labels	9
IPv6 Values and Labels	12
Status Bar	13
IP Library	14

Welcome to IPvX



IPvX

IPvX is a simple, easy-to-use, and intuitive IP calculator for both IPs.

- [Introduction](#)
- [User Interface](#)
- [IP Library](#)

License Information

Warning:

THIS COMPUTER PROGRAM IS PROTECTED BY COPYRIGHT LAW AND INTERNATIONAL TREATIES. UNAUTHORIZED REPRODUCTION OR DISTRIBUTION OF THIS PROGRAM, OR ANY PORTION OF IT, MAY RESULT IN SEVERE CIVIL AND CRIMINAL PENALTIES, AND WILL BE PROSECUTED TO THE MAXIMUM EXTENT POSSIBLE UNDER THE LAW.

Author:

Ron Maupin

Copyright:

IPvX © 2010 - 2022 by Ron Maupin

IP.pas © 2010 - 2022 by Ron Maupin

Velthuis.BigIntegers.pas © 2015,2016,2017 Rudy Velthuis

License:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notices, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Disclaimer:

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Credits:

Thanks to Rudy Velthuis for the BigInteger library to simplify IPv6 128-bit math. IPv6 addresses can be handled as an array of two 64-bit or four 32-bit, unsigned integers, but it is simpler and easier to be able to use 128-bit, unsigned integers which is what an IPv6 address is.

Introduction

- [About IPvX](#)
- [System Requirements](#)
- [Getting Help](#)

About IPvX

IPvX is a tool for IP address calculations (both IPv4 and IPv6) that began as a test for my IP library where there is one object class for IPv4 and another object class for IPv6. The IPv4 object properties are read and written as strings in the common dotted-decimal format (without leading zeroes in the octets), and the IPv6 object properties are written in any of the three [RFC 4291, Section 2.2](#) *conventional* text representations for IPv6, but IPv6 object properties are always read in the [RFC 5952, Section 4](#) *canonical* text representation for IPv6 addresses. Each object only stores two values: Address and Mask (IPv4 32-bit, unsigned integers, and IPv6 128-bit, unsigned integers). All the other properties of an object are calculated when they are read, and the read-write properties calculate and modify the IP object Address and/or Mask values when they are written. IPvX instantiates one object for each IP.

As IPvX evolved, various features, such as information about the IP addresses (cast, scope, and description) in a status bar, and a keypad (allowing mouse-only use of IPvX) were added to the application. The keypad was inspired by the [IPv6 Buddy](#).

The IP address status was informed by:

- [IANA IPv4 Special-Purpose Address Registry](#)
- [IANA IPv4 Multicast Address Space Registry](#)
- [IANA Internet Protocol Version 6 Address Space](#)
- [IANA IPv6 Special-Purpose Address Registry](#)
- [IANA IPv6 Multicast Address Space Registry](#)
- [various RFCs](#)

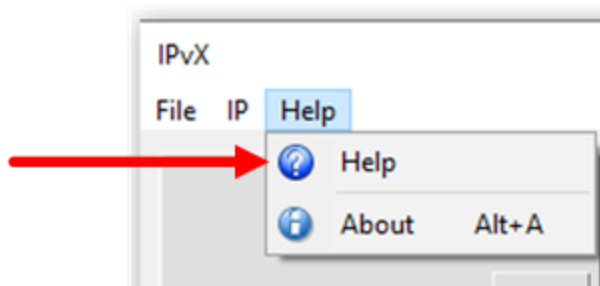
System Requirements

The IPvX recommended system configuration includes:

- Windows 10 or later
- 512 MB of RAM
- 5 MB of free disk space
- 1024x768 screen resolution or higher

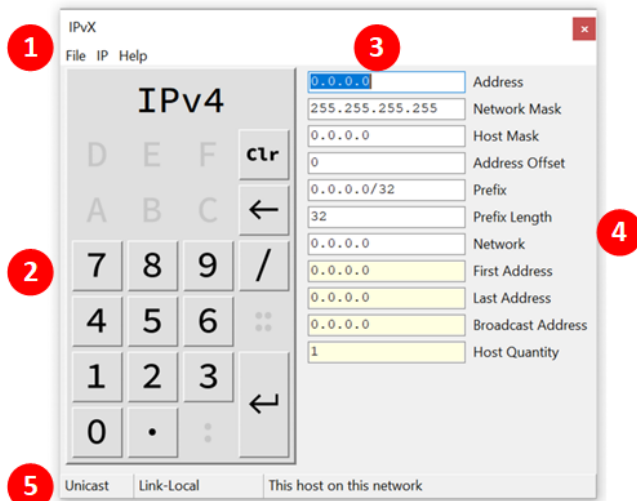
Getting Help

The IPvX help file is IPvX.chm. To launch it, either press the <F1> key for context sensitive help, or choose the **Help** menu item from the **Help** menu.



User Interface

The IPvX user interface:



1. [Menus](#)

- Reset the IP object of the currently chosen IP.
- Exit IPvX.
- Change the currently chosen IP.
- Launch this Help file.
- Launch the About form.

2. [Keypad](#)

- Facilitates mouse-only use of IPvX.
- Adapts its configuration for the currently chosen IP.

3. [Value Fields](#)

- Display the property values for the IP object of the currently chosen IP.
- Read-write value fields allow changing the properties for the IP object of the currently chosen IP.
- Adapt to the currently chosen IP.

4. [Labels](#)

- Identify the corresponding value fields.
- Adapt to the currently chosen IP.

5. [Status Bar](#)

- Displays the address cast.
- Displays the address scope.
- Displays the address description.
- Adapts to the currently chosen IP.

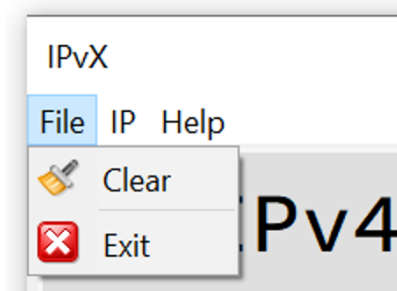
Menus

IPvX has three simple menus: **File**, **IP**, and **Help**.

- [File Menu](#)
- [IP Menu](#)
- [Help Menu](#)

File Menu

The **File** menu has two menu items: **Clear** and **Exit**.



Clear

Selecting the **Clear** menu item, or using the **<Esc>** key shortcut:

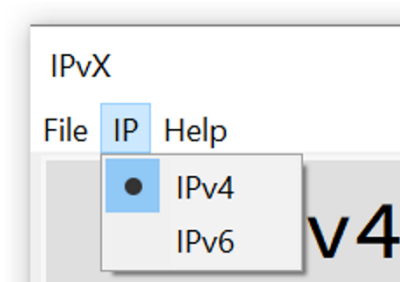
- Resets the currently selected IP object (sets the IP object's address to all-zeroes and the IP object's mask to all ones)
- Reads all the IPv4 object properties
- Updates the value fields with the IP object's property values
- Updates the status bar fields for the IP object's address

Exit

Selecting the **Exit** menu item, or using the **<Alt+X>** key shortcut, will close the IPvX application.

IP Menu

The **IP** menu has two menu items: **IPv4** and **IPv6**. The currently selected IP object is indicated by the dot image to the left of the menu item, and it is shown at the top of the keypad.



IPv4

Selecting the **IPv4** menu item, or using the **<Alt+4>** key shortcut:

- Selects the IPv4 object
- Reads all the IPv4 object properties
- Updates the value fields with the IPv4 object property values
- Updates the labels for the IPv4 object properties
- Updates the status bar fields for the IPv4 object address

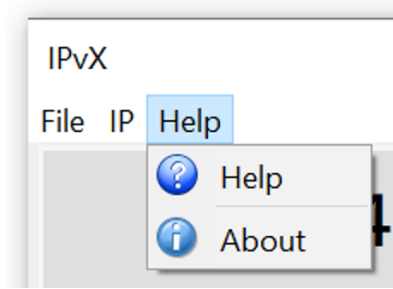
IPv6

Selecting the **IPv6** menu item, or using the **<Alt+6>** key shortcut:

- Selects the IPv6 object
- Reads all the IPv6 object properties
- Updates the value fields with the IPv6 object property values
- Updates the labels for the IPv6 object properties
- Updates the status bar fields for the IPv6 object address

Help Menu

The **Help** menu has two menu items: **Help** and **About**.



Help

Selecting the **Help** menu item, or pressing the **<F1>** key for context sensitive help, launches the Help file.

About

Selecting the **About** menu item, or using the **<Alt+A>** key shortcut, launches the About form that has information about the IPvX application.

Keypad

The keypad can be used to enter values into the value fields, allowing for mouse-only use of IPvX. The keypad adapts itself for the selected IP version, showing the IP version at the top of the keypad. Selecting IPv4 disables the keypad the IPv6-only keys. Selecting IPv6 enables all the keypad keys.

The keypad was inspired by the [IPv6 Buddy](#).

- [IPv4 Keypad](#)
- [IPv6 Keypad](#)

IPv4 Keypad

The IPv4 keypad shows **IPv4** at the top, and it enables just the keys necessary to use IPvX for IPv4 addressing:



0 - 9 Numeric Keys	The numeric keys are used to enter numeric digits in the value fields. Clicking on a numeric key is the equivalent of pressing a numeric key on the physical keyboard.
. Dot Key	The dot key is used to separate octets. Clicking on the dot key is the same as pressing the period or decimal key on the physical keyboard.
/ Slash Key	The slash key is used to enter a slash preceding a prefix length. Clicking on the slash key is the equivalent of pressing the slash key on the physical keyboard.
↵ Enter Key	The enter key is used to submit a value to the object property of the current IP object. Clicking on the enter key is the equivalent of pressing the enter key on the physical keyboard.
← Backspace	The backspace key is used to erase the digit to the left of the cursor. Clicking on the backspace key is the equivalent of pressing the backspace key on the physical keyboard.
clr Clear Key	The clear key is used to reset the current IP object to the default values. Clicking the key is the same as selecting the Clear menu item or pressing the escape key on the physical keyboard.

IPv6 Keypad

The IPv6 keypad shows **IPv6** at the top, and it enables just the keys necessary to use IPvX for IPv6 addressing:



0 - 9	Numeric Keys	The numeric keys are used to enter numeric digits in the value fields. Clicking a numeric key is the same as of pressing a numeric key on the physical keyboard.
A - F	Alphabetic Keys	The alphabetic keys are used to enter alphabetic digits in the value fields. Clicking an alphabetic key is the same as of pressing an alphabetic key on the physical keyboard.
.	Dot Key	The dot key is used to separate octets for mixed-mode addresses. Clicking the key is the same as pressing the period or decimal key on the physical keyboard.
:	Colon Key	The colon key is used to separate the hexadecimal fields (16-bit words) in the value fields. Clicking the key is the same as of pressing the colon key on the physical keyboard.
/	Slash Key	The slash key is used to enter a slash preceding a prefix length. Clicking the key is the same as of pressing the slash key on the physical keyboard.
::	Double-Colon Key	The double-colon key is used to enter two colons for the compressed format. Clicking the double-colon key is the same as pressing the colon key twice on the physical keyboard.
↵	Enter Key	The enter key is used to submit a value to the object property of the current IP object. Clicking on the enter key is the same as of pressing the enter key on the physical keyboard.
←	Backspace	The backspace key is used to erase the digit to the left of the cursor. Clicking on the backspace key is the same as of pressing the backspace key on the physical keyboard.
Ctrl	Clear Key	The clear key is used to reset the current IP object to the default values. Clicking the key is the same as selecting the Clear menu item or pressing the escape key on the physical keyboard.

Value Fields and Labels

The value fields and labels will vary, depending on the IP object selected. Each value field has a corresponding label to describe what the value field contains. Changing the IP object will change the visible value fields and their labels.

The value fields display the properties of the currently selected IP object. The values in each value field can be selected and copied into the Windows clipboard to be pasted into a different application. Most value fields are read-write fields that can be changed by the user to update the address and/or mask of the currently selected IP object, and the user can enter values into the read-write value fields with the keyboard, IPvX keypad, or paste values from the Windows clipboard.

- [IPv4 Values and Labels](#)
- [IPv6 Values and Labels](#)

IPv4 Values and Labels

The IPv4 object has 11 properties, the first seven of which are read-write properties, and the last four are read-only properties. The IPv4 object only stores an address and mask, and all the other properties are calculated when read, and the read-write properties modify the IPv4 object address and/or mask when written. Validation tests (format, range, etc.) are performed on values written to the read/write properties. Any invalid values result in an exception and a message dialog with the text message of the exception, and the values in the object remain unchanged. Exiting any read/write field in the application cause all the fields to be refreshed with the current values from all the object properties, and the three status fields (Cast, Scope, and Description) will also be updated.

0.0.0.0	Address
255.255.255.255	Network Mask
0.0.0.0	Host Mask
0	Address Offset
0.0.0.0/32	Prefix
32	Prefix Length
0.0.0.0	Network
0.0.0.0	First Address
0.0.0.0	Last Address
0.0.0.0	Broadcast Address
1	Host Quantity

Address

- **Read** - The text is created by converting the address value to text in the common IPv4 dotted-decimal notation.
- **Write** - The text is tested to determine that it is a valid IPv4 dotted-decimal value (*an optional CIDR notation network length may be included in the text*). The text is converted to an address value (*an optional network length is converted to a mask value*). The calculated address and existing mask (*or mask from an optional network length*) values are used to determine the full proposed network range that is tested to validate that the entire proposed network range falls entirely within a Unicast, Multicast, or Reserved range. Passing all validation tests replaces the existing address value (*and mask value for an optional network length*).

Network Mask

- **Read** - The text is created by converting the mask value to text in the common IPv4 dotted-decimal notation.
- **Write** - The text is tested to determine if it is a valid IPv4 dotted-decimal value. The text is converted to a mask value that is tested to check that it consists of consecutive, high-order 1 bits. The calculated mask and existing address values are used to determine the full proposed network range that is tested to validate that the entire proposed network range falls entirely within a Unicast, Multicast, or Reserved range. Passing all validation tests replaces the existing mask value.

Host Mask

- **Read** - The text is created by converting the inverse of the mask value to text in the common IPv4 dotted-decimal notation.
- **Write** - The text is tested to determine if it is a valid IPv4 dotted-decimal value. The text is converted to a mask value, inverted, and tested to check that it consists of consecutive, high-order 1 bits. The calculated mask and existing address values are used to determine the full proposed network range that is tested to validate that the entire proposed network range falls entirely within a Unicast, Multicast, or Reserved range. Passing all validation tests replaces the existing mask value.

Address Offset

- **Read** - The text is created by a bitwise **AND** of the address and inverse mask values and converting the result to integer text.
- **Write** - The text is converted to an integer value that is tested to verify that it is less than or equal to the inverse of the mask value. Passing all validation tests replaces the host portion of the address value with the calculated integer value.

Prefix

- **Read** - The text is created by a bitwise **AND** of the address and mask values to get the network address value that is converted to text in the common IPv4 dotted-decimal notation, a slash ("/") is appended, then the prefix length is created by counting the number of consecutive, high-order 1 bits in the mask value and converting the number of bits to integer text that is appended.
- **Write** - The text is tested to determine if it is a valid IPv4 dotted-decimal value, including a slash and valid prefix length (0 to 32). The network address part of the text is converted to an address value. The prefix length part of the text is converted to an integer value from which a mask value is calculated by setting the integer value number of high-order 1 bits in the value. A bitwise **AND** of the the calculated address and calculated mask values to arrive at the network address value that is tested to see if it matches original calculated address value. The calculated network address and mask values are used to determine the full proposed network range that is tested to validate that the entire proposed network range falls entirely within a Unicast, Multicast, or Reserved range. Passing all validation tests replaces the network portion of the existing address value and the existing mask value.

Prefix Length

- **Read** - The text is created by counting the number of consecutive, high-order 1 bits in the mask value that is converted to integer text.
- **Write** - The text is converted to an integer value and tested to see that it falls in the valid range (0 to 32). The integer value is converted to a mask value by setting the integer value number of high-order 1 bits in the value. The calculated mask and existing address values are used to determine the full proposed network range that is tested to validate that the entire proposed network range falls entirely within a Unicast, Multicast, or Reserved range. Passing all validation tests replaces the existing mask value.

Network

- **Read** - The text is created by a bitwise **AND** of the address and mask values to get the network address value that is converted to text in the common IPv4 dotted-decimal notation.
- **Write** - The text is tested to determine if it is a valid IPv4 dotted-decimal value. The text is converted to a an address value. A bitwise **AND** of the the calculated address and existing mask to arrive at the network address value is tested to see if it matches original calculated address value. The calculated address and existing mask values are used to determine the full proposed network range that is tested to validate that the entire proposed network range falls entirely within a Unicast, Multicast, or Reserved range. Passing all validation tests replaces the network portion of the existing address value.

First Usable Address

- **Read** - The text is created by a bitwise **AND** of the address and mask values, to which 1 is added if the network is in the Unicast range and the network length is less than 31, then converted to text in the common IPv4 dotted-decimal notation.

Last Usable Address

- **Read** - The text is created by a bitwise **AND** of the address and mask values, to which the inverse of the mask value is added, from which 1 is subtracted if the network is in the Unicast range and the network length is less than 31, then converted to text in the common IPv4 dotted-decimal notation.

Broadcast Address


- **Read** - The text is created by a bitwise **AND** of the address and mask values, to which the inverse of the mask value is added, then if the network is in the Unicast range, converted to text in the common IPv4 dotted-decimal notation, or "N/A" if the network is in a Multicast or Reserved range.

Host Quantity

- **Read** - The text is created by a bitwise **AND** of the address and inverse mask values, to which 1 is added, then is converted to integer text.

IPv6 Values and Labels

The IPv6 object has eight properties, the first five of which are read-write properties, and the last three are read-only properties. The IPv6 object only stores an address and mask, and all the other properties are calculated when read, and the read-write properties modify the IPv6 object address and/or mask when written. Validation tests (format, range, etc.) are performed on values written to the read/write properties. Any invalid values result in an exception and a message dialog with the text message of the exception, and the values in the object remain unchanged. Exiting any read/write field in the application cause all the fields to be refreshed with the current values from all the object properties, and the three status fields (Cast, Scope, and Description) will also be updated.

	Address
0	Address Offset
::/128	Prefix
128	Prefix Length
::	Network
::	First Address
::	Last Address
1	Host Quantity

Address

- **Read** - The text is created by converting the address value to text in the RFC 5952 IPv6 canonical text representation.
- **Write** - The text is tested to determine that it is a valid RFC 4291 IPv6 conventional text representation (*an optional CIDR notation network length may be included in the text*). The text is converted to an address value (*an optional network length is converted to a mask value*). The calculated address and existing mask (*or mask from an optional network length*) values are used to determine the full proposed network range that is tested to validate that the entire proposed network range falls entirely within a Unicast, Multicast, or Reserved range. Passing all validation tests replaces the existing address value (*and mask value for an optional network length*).

Address Offset

- **Read** - The text is created by a bitwise **AND** of the address and inverse mask values and converting the result to integer text.
- **Write** - The text is converted to an integer value that is tested to verify that it is less than or equal to the inverse of the mask value. Passing all validation tests replaces the host portion of the address value with the calculated integer value.

Prefix

- **Read** - The text is created by a bitwise **AND** of the address and mask values to get the network address value that is converted to text in the RFC 5952 IPv6 canonical text representation, a slash ("/") is appended, then the prefix length is created by counting the number of consecutive, high-order 1 bits in the mask value and converting the number of bits to integer text that is appended.
- **Write** - The text is tested to determine if it is a valid RFC 4291 IPv6 conventional text representation, including a slash and valid prefix length (0 to 128). The network address part of the text is converted to an address value. The prefix length part of the text is converted to an integer value from which a mask value is calculated by setting the integer value number of high-order 1 bits in the value. A bitwise **AND** of the the calculated address and calculated mask values to arrive at the network address value that is tested to see if it matches original calculated address value. The calculated network address and mask values are used to determine the full proposed

network range that is tested to validate that the entire proposed network range falls entirely within a Unicast, Multicast, or Reserved range. Passing all validation tests replaces the network portion of the existing address value and the existing mask value.

Prefix Length

- **Read** - The text is created by counting the number of consecutive, high-order 1 bits in the mask value that is converted to integer text.
- **Write** - The text is converted to an integer value and tested to see that it falls in the valid range (0 to 128). The integer value is converted to a mask value by setting the integer value number of high-order 1 bits in the value. The calculated mask and existing address values are used to determine the full proposed network range that is tested to validate that the entire proposed network range falls entirely within a Unicast, Multicast, or Reserved range. Passing all validation tests replaces the existing mask value.

Network

- **Read** - The text is created by a bitwise **AND** of the address and mask values to get the network address value that is converted to text in the RFC 5952 IPv6 canonical text representation.
- **Write** - The text is tested to determine if it is a valid RFC 4291 IPv6 conventional text representation. The text is converted to an address value. A bitwise **AND** of the the calculated address and existing mask to arrive at the network address value is tested to see if it matches original calculated address value. The calculated address and existing mask values are used to determine the full proposed network range that is tested to validate that the entire proposed network range falls entirely within a Unicast, Multicast, or Reserved range. Passing all validation tests replaces the network portion of the existing address value.

First Network Address

- **Read** - The text is created by a bitwise **AND** of the address and mask values that is converted to text in the RFC 5952 IPv6 canonical text representation.

Last Network Address

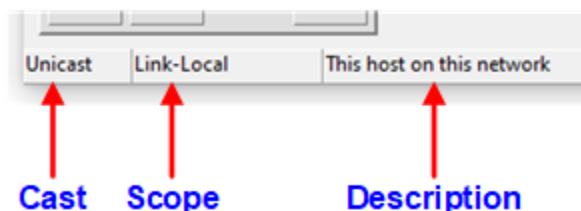
- **Read** - The text is created by a bitwise **AND** of the address and mask values, to which the inverse of the mask value is added, then converted to text in the RFC 5952 IPv6 canonical text representation.

Host Quantity

- **Read** - The text is created by a bitwise **AND** of the address and inverse mask values, to which 1 is added, then is converted to integer text.

Status Bar

The Status Bar has three fields that describe the IP address of the currently selected IP object. The status bar fields are populated by IPvX, not the IP object properties.



Cast

The cast field describes the cast of the address in the currently selected IP object:

- Unicast addresses can be used as both source and destination addresses, and are destined for a single host.
- Multicast addresses are group addresses that can only be used as destination addresses, and are destined only for hosts subscribed to the multicast group

- Broadcast addresses are group addresses that can only be used as destination addresses, and are destined for all hosts on a network (not used in IPv6)
- Reserved addresses cannot be used

Scope

There are various address scopes, depending on the IP version and cast, that were gleaned from the various IANA documents and IETF RFCs.

Description

The address descriptions are more-or-less official descriptions that were gleaned from the various IANA documents and IETF RFCs.

IP Library

The core of IPvX is the IP library. IPvX began as a test of the IP library. The IP library was conceived as a way to perform IP math while reading and writing human-readable strings. IP math cannot properly be performed with the string representation of IP addresses, and programmers try to do that with very bad results.

There are two object class definitions in the IP library: one each for IPv4 and IPv6. The object properties are all string properties, but the strings are converted to unsigned integers (32-bit for IPv4 and 128-bit for IPv6) to correctly perform IP math, then back to strings when read. Each IP object only stores two values: Address and Mask as unsigned integers for the IP version of the object. Using a string that is not valid for an IP object property will throw an `EIPError` exception with a description relevant to the error that caused the exception.

Each IP class also includes a few useful class functions:

- **IsUnicast** (IPv4/IPv6) returns **True** if the IP address string parameter is in a Unicast address range, otherwise it returns **False**.
- **IsMulticast** (IPv4/IPv6) returns **True** if the IP address string parameter is in a Multicast address range, otherwise it returns **False**.
- **IsReserved** (IPv4/IPv6) returns **True** if the IP address string parameter is in a Reserved address range, otherwise it returns **False**.
- **IsInRange** (IPv4/IPv6) returns **True** if the IP address string parameter is in the address range of the IP prefix parameter, otherwise it returns **False**.
- **Expand** (IPv6) returns a fully expanded (eight four-digit, colon separated, hexadecimal fields) IPv6 address string from the IPv6 address string parameter.
- **Compress** (IPv6) returns a compressed (RFC 5952) IPv6 address string from the IPv6 address string parameter.

IPvX instantiates one IPv4 object, and one IPv6 object. Each of the value fields in IPvX simply reads the corresponding IP object property, and changing one of the read-write fields tries to change the corresponding IP object property, returning a dialog with the text of the `EIPError` if the entered text causes an exception in the IP object.

The IP library does not include the cast, scope or descriptions reported by IPvX. Those values are calculated in IPvX by using the `IsInRange` class function of each IP class to test if the address stored in the IP object is within specific ranges that meet the IANA and IETF definitions for the cast, scope, and description.