nCircle IP360™ v6.8.3

API Guide

nCircle, Inc.
101 Second St., Suite 400
San Francisco, CA 94105

Phone: +1 (888) 464-2900
Fax: +1 (415) 625-5982
Email: support@ncircle.com

Copyright Notice

Copyright © 2010 nCircle Network Security

All rights reserved. Printed in the U.S.A.

All specifications regarding products in this document are subject to change without notice. All statements, information, and recommendations in this document are believed to be accurate but are presented without warranty of any kind, express or implied. Users must take full responsibility for their application of any products discussed in this document. Except as permitted by such license, no part of this document may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without prior written permission of nCircle.

The content of this document is furnished for informational use only, is subject to change without notice, and is not to be construed as a commitment by nCircle. nCircle assumes no responsibility for liability for errors or inaccuracies in this manual.

Trademark Notification

nCircle™, IP360™, Reflex Testing™, Ontology™, FactoredReasoning™, Featureprinting™, Focus™, Discriminant Analysis™, nCircle Security Intelligence Hub™, nCircle Topology Risk Analyzer™, nCircle Configuration Compliance Manager™, nCircle PCI Scan Service™, and nCircle nTellect™ are all trademark and service marks of nCircle Network Security.

Internet Explorer is a registered trademark of Microsoft Corp.

Netscape, Netscape Communicator, and Netscape Navigator are registered trademarks of Netscape Communications Corporation in the United States and other countries.

SSH Secure Shell is a trademark of SSH Communications Security.

Cisco is a registered trademark of Cisco Systems Inc. and/or its affiliates in the United States and certain other countries.

BMC Software, the BMC Software logos and all other BMC Software product or service names are registered trademarks or trademarks of BMC Software, Inc.

Crystal Reports is a registered trademark of Business Objects.

ActiveState and ActivePerl are registered trademarks of ActiveState.

iDEFENSE is a registered trademark of iDEFENSE Inc., a Verisign Company.

Macromedia Flash Player is a registered trademark of Adobe.

Symantec and DeepSight are registered trademarks of Symantec Corporation.

All other product names mentioned herein are trademarks or registered trademarks of their respective owners.

This information is strictly confidential and for the expressed use by nCircle customers for the use of the nCircle family of products. The information provided herein is to be considered nCircle Confidential Information and its use shall be governed by the license and non-disclosure agreements in place between nCircle and the recipient organization. Distribution beyond the intended customer recipient, or to third parties, is expressly forbidden.

February 10, 2010

Contents

Chapter 1 Introduction to the IP360 API

The nCircle IP360 API enables users of the system to programmatically command and control the functionality of IP360. The API allows users to easily remotely control IP360 and integrate it with third-party products.

Overview of the Guide

The nCircle IP360 API Guide is organized as follows:

The **"Introduction to the IP360 API"** chapter provides general rules for using the API.

The **"The IP360 Object Model"** chapter describes the IP360 object model and how the API (and consequently the API client and user) manipulates the object model to perform IP360 tasks.

The **"IP360 API Reference"** chapter is a reference guide for all object classes in the API. It describes all the IP360 objects and

functions available through the API. This reference information provides the necessary detail for constructing API requests.

The **"IP360 API Python Library"** chapter introduces and illustrates the use of the nCircle IP360 Python Library, which API users can take advantage of to more easily write programs to talk to the API.

The **"API Error Handling"** chapter explains how the API handles errors and enumerates the fault codes it uses.

The **"Sample Code for Basic API Tasks"** chapter provides sample code for basic IP360 tasks.These samples should give API users a stepping-off point for using the API for their own IP360 and business needs.

The **"XML-RPC Protocol Details"** appendix describes the API's XML-RPC implementation, shows users how they can construct clients to interact with the IP360 API, and provides examples of API requests in XML.

Functions Available Through the API

IP360 API users can perform the following IP360 tasks:

- Users and user groups (create, modify, delete)
- Roles (delete, view users in)
- Customers (set red score threshold)
- Credentials (create, bind, delete, modify)
- Global IP Exclude list (view)
- Appliances (Device Profiler, Threat Monitor, nTellect) (modify)
- Networks and network groups (create, modify, delete)
- Scan profiles (create, modify, delete, read)
- Scan configurations (create, delete)
- Scans (Audits) (start and control)
- ASPL (download the latest ASPL data)
- External Ticketing (verification scans)

For detailed reference information on API classes of IP360 objects and their associated functions (methods), see "IP360 API Reference" on page 12.

User Interface and API Sessions

Users can use IP360 through the API just as they use it through the UI, limited only by the availability of functions through the API. A single user is permitted one UI session simultaneous with multiple simultaneous API sessions.

API Compatibility

Customers should ensure that their API clients are compatible with the version of the API protocol they are interacting with. The API has the same version as the overall IP360 product, but the API protocol has an independent version number, consisting of a major version and a minor version.

The major version number is incremented when a non-backwards-compatible change is introduced into the API. For example, API protocol 2.x is *not* expected to work with code written against API protocol 1.x. The minor version number is incremented when a backwards-compatible change is introduced. For example, API protocol 1.1 is expected to work with code written against API protocol 1.0.

XML-RPC API Interface

The IP360 API uses an HTTPS XML-RPC interface. Knowing the details of the XML-RPC implementation is not necessary because users can use Python libraries (including the nCircle IP360 Python library) or other languages such as Adobe Flex to interact with the API. More information on the XML-RPC implementation, however, is available in the appendix, "XML-RPC Protocol Details" on page 85.

Getting Started with the IP360 API

These directions for getting started with the IP360 API assume that users have already installed and used IP360 6.8.3.

To connect for the first time with the IP360 API:

1. (*Optional*) Create a new user account to serve as the API user.
2. Assign this user the privileges necessary to carry out the operations to be scripted. This user's password will be marked "expired" by the system on creation, so it is necessary to log in once as this user and change the password before it can be used with the API. The API will reject an attempt to login by a user with an expired password.
3. Obtain an implementation of the XML-RPC protocol for the desired programming language. See http://www.xmlrpc.com/directory/1568/implementations or choose from this list:
   - PHP: XML-RPC for PHP (http://phpxmlrpc.sourceforge.net/)
   - Java: Apache XML-RPC (http://ws.apache.org/xmlrpc/)
   - Perl: RPC: XML module (http://search.cpan.org/~rjray/RPC-XML-0.58/)
   - Python: xmlrpclib module (included with the Python distribution)
   - Adobe Flex: as3-rpclib (http://code.google.com/p/as3-rpclib/)
4. Write a simple program to verify connectivity to the API.
   Perl example:

```
require RPC::XML;
require RPC::XML::Client;

my $client = RPC::XML::Client->new('https://ip360.example.com/api/');
my $cookie = $client->simple_request('login', 1, 0,
                                     'user@example.com', 'password');
my $result = $client->simple_request('call', $cookie, 'SESSION',
                                     'getUserObject', {});
```

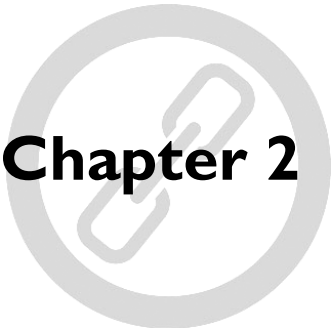
```
print "My user object is: $result\n";
```

Python example (using xmlrpclib):

```
import xmlrpclib

server = xmlrpclib.ServerProxy('https://ip360.example.com/api/')
cookie = server.login(1, 0, 'user@example.com', 'password')
result = server.call(cookie, 'SESSION', 'getUserObject', {})
print "My user object is: %s" % (result)
```

Python example (using the nCircle IP360 API library):

```
import IP360.API

session = IP360.API.Session('ip360.example.com')
result = session.getUserObject()

print "My user object is: %s" % (result)
```



Chapter 2 The IP360 Object Model

IP360 functions and stored data are represented in the IP360 Object Model. This chapter describes the object model and how it is implemented by the API. Readers will better understand the API reference information (in Chapter 3) by first reading this chapter.

There are two kinds of objects represented by the model: class objects and instance objects.

Note: Examples in this chapter are as implementation language-neutral as possible. Different languages may have different requirements for passing text and integers to the API. Be sure to consult documentation for the correct implementation language.

The Session Object

The Session Object is a special IP360 API class object that provides access to information in the API not represented by any particular class, that is, the class objects themselves. It always has the object reference `SESSION`.

Example. To get a list of the names of the classes provided by the IP360 API, perform a method call with the parameters:

Object	`SESSION`
Method	`getClassList`
Arguments	None

This will return a list of strings:

```
"User", "Network", "ScanProfile", "Appliance", ...
```

To get the object reference for the User class, perform a method call with the parameters:

Object	`SESSION`
Method	`getClassByName`
Arguments	`name`="User"

This will return the string:

```
"class.User"
```

For complete details on the methods implemented by the `SESSION` object, see "SESSION Object Methods" on page 71. For more information on object references, see "Object References" on page 7.

Class Objects

A class object provides a generic representation of some class of data in the system (for example, networks). Users can call "class methods" on a class object to discover what data attributes an instance of the class has and what methods can be called on an instance of the class. Users can also use class methods to look up existing instances of the class and to create new ones.

Class objects have an object reference that looks like:

```
class.<classname>
```

For example, the Device Profiler class is referenced by:

```
class.DP
```

and the Network class by:

```
class.Network
```

Example. To get a list of the names of the attributes defined by the Network class, perform a method call with the parameters:

Object	`class.Network`
Method	`getAttributes`
Arguments	None

This will return a list of strings:

```
"NetBIOSPrimary", "active", "assetValue", "customer", "deleted",
"id", "include", "name", "notes", "static"
```

Instance Objects

An instance object represents a particular entity in the system (for example, a single network). Users can call methods on the instance object to retrieve the values of its attributes, to change the values of its attributes, or to perform other specific actions particular to the class of the object.

Instance objects have an object reference that looks like:

```
<classname>.<id>
```

For example, a user instance might have the object reference `User.6`, and a network instance might have the object reference `Network.44`.

Example. To change the password of the user with object reference `User.6` to "newpassword," perform a method call with the parameters:

Object	`User.6`
Method	`setPassword`
Arguments	`password: "newpassword"`

Object References

Every object in the IP360 Object Model has an "object reference" ("objref"), a string unique to that object that is used to refer to that object. Object references are returned by some methods and are passed as arguments to others. API clients may perform a string comparison between two object references returned by the API to determine if they refer to the same object.

In practice, object references should continue to be valid indefinitely. Strictly speaking, though, the IP360 API guarantees that an object reference will continue to refer to the same object in the system until the next time the IP360 software is upgraded.

nCircle cautions API client authors against manually constructing object reference strings. The references are transparent for ease of understanding and client diagnostics, but to ensure compatibility with future versions of the API, customers should always request them from the API using the appropriate methods. The `SESSION` object is an exception because it is explicitly defined as part of the IP360 API protocol.

Method Arguments

Arguments to API methods are named. A method call specifies a list of name-value pairs (represented by an XML-RPC "struct"), where each pair consists of the name of an argument and the value being passed for that argument. Some method arguments are optional, as designated in the "IP360 API Reference" on page 12. These arguments may be omitted from the argument list. Mandatory arguments must be included in the argument list, or else the method call will fail. The method call will also fail if it is passed an argument that is not supported by the method.

Search Queries

IP360 API search queries are written in a subset of the SQL language. A search operation is like a `SQL SELECT` query against a set of objects, and the search query is the equivalent of the `WHERE` clause.

The basic search query syntax makes it possible to compare object attributes to known values or patterns. In addition, basic search queries can be combined into boolean expressions using `AND`, `OR`, and parentheses. A search query is a string.

All of the query values listed in this section are examples of values for the `query` argument in the common class method `search`. See "search" on page 69 for reference information.

Basic Search Queries

A basic search query has the form:

```
query: "<attribute name> = <value>"
```

or

```
query: "<attribute name> ~ <POSIX extended regular expression>"
```

or

```
query: "<attribute name> LIKE <SQL LIKE expression>"
```

where:

- `<attribute name>` is the name of a class attribute (specific classes' attributes are enumerated in Chapter 3, "IP360 API Reference" on page 12)
- `<value>` is either a literal integer value (for example, 123) or a SQL-quoted string value (for example, `'abc'`)
- `<SQL LIKE expression>` is a quoted string where "%" is a wildcard that substitutes for any substring
- `<POSIX extended regular expression>` is a quoted regular expression as defined in IEEE Std 1003.2 ("POSIX.2").

Boolean Operators. A query can use the following operators (including the ones listed above): = (equal to), < (less than), <= (less than or equal to), > (greater than), >= (greater than or equal to), ~ (regular express match), LIKE (SQL LIKE expression match).

Example. Following are example query values for the search method:

```
query: "name = 'San Francisco Network'"
query: "endTime > 1146774911"
```

```
query: "name ~ '(San Francisco|New York|Singapore) Network'"
query: "name LIKE '%North America%'"
```

Boolean Search Queries

A boolean query expression has the form:

```
query: <basic query>
```

or

```
query: "<expr>"
```

or

```
query: "<expr> AND <expr>"
```

or

```
query: "<expr> OR <expr>"
```

or

```
query: "NOT <expr>"
```

where:

- `<basic query>` is a basic search query as defined above.
- `<expr>` is any boolean query expression, *including* boolean attributes of objects, for example, the "active" attribute for a Network object, or the "deleted" attribute for a Scan Profile object.

Example.

In the User class, this could be used as a query value:

```
query: "firstName = 'Bob' OR phoneNumber LIKE '%555-1212'"
```

In the Audit class, this could be used as a query value:

```
query: "network = 'Network.12' AND (status = 4 OR status = 5 OR
                                    status = 6)"
```

In the Network class, this could be used as a query value:

```
query: "NOT active"
```

Search Result Formats

The IP360 API supports two search result formats: a list and a table. The format is specified by the `format` argument to those API methods that return search results.

Search Result Type "list". A "list" search result is a list of strings, where each string is an object reference. List search results are relatively inexpensive to generate and transmit over the network, as only a few bytes are required for each search result. However, if the API client needs to retrieve attribute information for the objects, the API client must later perform an individual request for each object reference.

Search Result Type "table". A "table" search result contains the object references of the matching objects along with the basic attribute values of those objects. (Compound attributes, for example, the "includes" or "excludes" list attributes of the Network class, are not included in the

table.) The table search result is formatted as an XML-RPC struct with two named members: "columns" and "table." The "columns" member is a list of strings containing the names of the attributes represented by each column in the table. The "table" member is a list of lists, with one list for each row in the table. The first column in every search result table is named "REFERENCE", and in each row it contains the object reference for the object described by that row in the table.

For example, search results from the ScanConfiguration class might look like:

```
columns: [ "REFERENCE", "id", "network", "scanProfile", "dp", "active" ]
table: [ [ "ScanConfiguration.1", 1, "Network.3", "ScanProfile.8", "DP.1",
           1 ]
         [ "ScanConfiguration.2", 2, "Network.4", "ScanProfile.8", "DP.1",
           0 ]
       ]
```

The XML-RPC result would be:

```
<struct>
  <member>
    <name>columns</name>
    <value>
      <array>
        <value><string>REFERENCE</string></value>
        <value><string>id</string></value>
        <value><string>network</string></value>
        <value><string>scanProfile</string></value>
        <value><string>dp</string></value>
        <value><string>active</string></value>
      </array>
    </value>
  </member>
  <member>
    <name>table</name>
    <value>
      <array>
        <value>
          <array>
            <value><string>ScanConfiguration.1</string></value>
            <value><int>1</int></value>
            <value><string>Network.3</string></value>
            <value><string>ScanProfile.8</string></value>
            <value><string>DP.1</string></value>
            <value><int>1</int></value>
          </array>
        </value>
        <value>
          <array>
            <value><string>ScanConfiguration.2</string></value>
            <value><int>2</int></value>
            <value><string>Network.4</string></value>
            <value><string>ScanProfile.8</string></value>
            <value><string>DP.1</string></value>
            <value><int>0</int></value>
          </array>
        </value>
      </array>
    </value>
  </member>
</struct>
```

Enumerated Data Types

Enumerated data types represent a selection from a set of choices. The actual value is represented as an integer when passed to or from the API, and the API provides the means to translate these integer values to human-meaningful strings.

Each enumerated type has a different set of supported values. For example, the AuditStatus enumeration has these values:

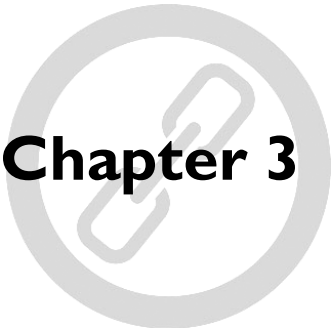
1: In Progress
2: Failed
3: Cancelled
4: Finished
5: Paused
6: Auto-Paused
7: Suspended

To perform a search for all "In Progress" audits, perform a method call with the parameters:

Object	`class.Audit`
Method	`search`
Arguments	`query: 'status = 1'`

To retrieve a list of supported values for any enumeration, perform a method call with the parameters (see "getEnumValues" on page 72 for more information):

Object	`SESSION`
Method	`getEnumValues`
Arguments	`name:` <enunumeration name>



Chapter 3 IP360 API Reference

This chapter provides an alphabetical list of IP360 API classes, their attributes, their methods, and their class methods. At the end of the chapter are described methods common to all classes and to all instances and methods implemented by the `SESSION` object.

Readers should review both "The IP360 Object Model" on page 5 and "Explanation of Terms" on page 15 to better understand the reference information in this chapter.

For more information about the UI analogs to all the objects and tasks described in this chapter, see the IP360 Administrator Guide.

Note: Scans are controlled through the DP class. See "Class: DP" on page 37.

Overview of IP360 API Classes

The IP360 API supports the following classes:

Table 1. Supported API Classes

Class Name	Page	Description
Appliance	16	This abstract class represents appliances (Device Profilers, Threat Monitors, and nTellect) connected to the IP360 system. See the DP, TM, and IPSnTellect classes for appliance-specific information.
ASPL	17	This singleton class provides access to the IP360 ASPL database.
Audit	18	This class represents the scan records (“audits”) in the IP360 system.
Credential	20	This abstract class represents credentials (SMB, SSH password, SSH key, and SNMP). See the CredSMB, CredSSHPassword, CredSSHKey, CredSNMP, CredWebForm and CredWebHTTP classes for credential-specific information.
CredSMB	20	This class represents SMB credentials used by IP360. See the abstract Credential class for general credential information.
CredSSHPassword	23	This class represents SHH password credentials used by IP360. See the abstract Credential class for general credential information.
CredSSHKey	25	This class represents SSH key credentials used by IP360. See the abstract Credential class for general credential information.
CredSNMP	27	This class represents SNMP credentials used by IP360. See the abstract Credential class for general credential information.
CredWebForm	29	This class represents credentials for web forms requiring authentication.
CredWebHTTP	32	This class represents credentials for applications requiring HTTP authentication.
Customer	35	This class represents customer records in the IP360 system.
DP	37	This class represents the Device Profiler appliances connected to the IP360 system. See the abstract Appliance class for general appliance information.
ExternalTicketing	n/a	This singleton class provides access to the IP360 external ticketing apparatus. For more information, consult the “nCircle IP360 Custom Ticketing Guide.”

Table 1. Supported API Classes

Class Name	Page	Description
GlobalIpExclude	40	This class represents the Global Exclude lists defined in IP360, which enumerate IP addresses to be excluded from all scanning.
IPSnTellect	42	This class represents the nTellect appliances connected to the IP360 system. See the abstract Appliance class for general appliance information.
Network	43	This class represents the networks configured in the IP360 system. A network is a collection of IP space that can be specified as a scan target for a Device Profiler or a monitoring target for an IPS nTellect (or for a Threat Monitor, if it is part of the IP360 deployment).
NetworkGroup	47	This class represents the network groups configured in the IP360 system. A network group is a collection of networks and network groups organized for common reporting and administration.
Role	51	This class represents roles in the IP360 system.
ScanConfiguration	52	This class represents the scan configurations configured in the IP360 system. A scan configuration specifies a network to scan, a Device Profiler to perform the scan, and a scan profile to instruct the Device Profiler how to perform the scan.
ScanProfile	53	This class represents the scan profiles configured in the IP360 system. A scan profile is a set of instructions to the Device Profiler specifying which scan operations to perform during a scan of a network.
ThreatZoneGroup	55	This class represents Topology Threat Zone Groups. Threat Zone Groups are used in Topological Risk Analysis to define perspectives from which to evaluate line of sight to network hosts and the expected threat level from each of those perspectives.
TM	57	This class represents the Threat Monitor appliances connected to the IP360 system. See the abstract Appliance class for general appliance information.
TopoAnalysis	58	This class generates a Topological Risk Analysis report.
TopoConfiguration	61	This class configures TRA data for export.
User	63	This class represents the users in the IP360 system.
UserGroup	66	This class represents user groups in the IP360 system.

Explanation of Terms

The following are terms and concepts used in the description of a class.

"Singleton" Classes. A "singleton" class has no instances. It simply provides access to parts of the IP360 sysem that are not broken into smaller parts (as far as the API is concerned).

"Read Only" Attributes. An attribute that is "Read Only" cannot have its value changed via the API.

"Sortable" Attributes. A "sortable" attribute may be used as the parameter by which to sort the results in a `search` or `fetch` query.

"Enum" Data Type. A value with an "Enum" (enumerated) data type has a value that comes from a defined set of choices. For more information, see "Enumerated Data Types" on page 11.

"Instance Methods" Vs. "Class Methods". "Instance methods" are to be used on instances of a class object; "class methods" are to be used on a class object itself. (See "Class Objects" on page 6 and "Instance Objects" on page 7 for more information on the distinction between classes and instances and between class methods and instance methods.)

Method Access Rights. "Minimum required access rights" are specified for each method. These are the same as the access rights required to perform the same task through the IP360 UI. Access rights are fully explained in the "nCircle IP360 Administrator Guide."

Return Type of "Mixed". A "Mixed" return type indicates that a method can return any API data type. In practice, this type is usually specified when the return data type is a complex structure, like the table of tickets returned by `ExternalTicketing:getTickets`.

`Clear` **and** `Set` **Methods.** `clear` methods delete ("clear") the value of an instance's attribute to make way for a new value to be specified ("set").

Data Types

The IP360 API uses several different data types to represent and structure information passed to and from the API.

Table 2. API Data Types

Data Type	Explanation
Integer	An integer value between -2147483648 and 2147483647. Encoded as an XML-RPC <int> value.

Integer (for times)	The number of non-leap seconds since the epoch (1 January 1970, 00:00.00, GMT).
	To convert this value to a human-readable time, in Python:
	`> import time`
	`> print time.ctime(998091243)`
	`Fri Aug 17 16:34:03 2001`
	
	With an `audit` object using the IP360 API Python library:
	`print 'Audit started at %s and ended at %s.' % \`
	`(time.ctime(audit.startTime),`
	`time.ctime(audit.endTime))`

String	A variable-length UTF-8 string. Normally encoded as an XML-RPC <string> value, it can be encoded as an XML-RPC <base64> value by the IP360 API when this is a more space-efficient way to represent the string, for example, when it consists of XML test that would have otherwise required XML-escaping of many <, >, and & characters.
List	A variable-length list of data elements. Encoded as an XML-RPC <array> value.
Dictionary	A variable-length associative array, that is, an array consisting of paired keys and values. The keys are always strings, and the values may each be of any data type. Encoded as an XML-RPC <struct> value.
Table	A tabular representation of data elements arranged into rows and columns. Represented by a two-item XML-RPC struct, where the "columns" key is associated with a list of strings naming the columns, and the "table" key is associated with an array of arrays, one for each row in the table. Each array within the "table" array contains the same number of elements as the "columns" array, each element corresponding with the column named in the same position of the "columns" array. (See "Search Result Type "table"" on page 9 for more information and an example.)

Class: Appliance

This abstract class represents appliances (Device Profilers, Threat Monitors, and nTellects) connected to the IP360 system. This class defines attributes and methods common to all appliances, but the API will never return a reference to an instance of this class. A Device Profiler appliance will be represented by a reference to an instance of the DP class, and an nTellect and a Threat Monitor similarly by a reference to an instance of the IPSnTellect class and ThreatMonitor

class, respectively. The Appliance class is mainly useful for performing queries across all types of appliances.

Attributes

Table 3. "Appliance" Class Attributes

Name	Description	Type	Read Only	Sortable
id	system internal identifier the appliance	integer	Y	Y
customer	object reference to the customer to whom this appliance is assigned	objref	Y	Y
type	the type of appliance	enum(ApplianceType)	Y	N
hardwareType	the appliance's type of hardware	enum(HardwareType)	Y	N
name	the appliance's user-specified name	string	Y	N
IPAddress	the appliance's IP address	string	Y	N
Key	the appliance's auth key	string	Y	N

Instance Methods

None.

Class Methods

None.

Class: ASPL

This singleton class provides access to the IP360 ASPL database.

Attributes

None.

Instance Methods

None.

Class Methods

getASPLVersion

Return the current version of the ASPL database.
No access rights required.
No arguments.
Returns = the current version of the ASPL database (string). Detects the modular ontology versions installed and generates an ASPL version as shown in the following examples:

- base-1-248 + nonpci-1-248 = M-248
- base-1-248 + spider-1-248 + pci-1-248 = N-248
- base-1-248 + nonpci-1-248 + spider-1-248 + web-1-248 = W-248

If a combination of packages seem invalid then the "M-<base package number>" is returned.

If the ontology packages are at different revision number then the base ontology package's revision number is returned. For example, N-249 is returned if the packages installed are base-1-249,spider-1-248,pci-1-248.

getModularASPLVersion

Return the modular ontology version of the ASPL database.
No access rights required.
No arguments.
Returns = the current version of the ASPL database(s) (string) using the modular ontology naming convention. Multiple versions are separated by commas, for example, “base-1-248, spider-1-248, pci-1-248.”

getASPLXML

Return an XML export of the current ASPL database.
No access rights required.
Returns = a link to an XML export of the current ASPL database (link)

Table 4. “getASPLXML” Arguments

Name	Description	Type	Optional
language	language of exported descriptions. Valid values are “en” for English and “jp” for Japanese. The default is “en.”	string	Y

Class: Audit

This class represents the scan records (“audits”) in the IP360 system.

Attributes

Table 5. "Audit" Class Attributes

Name	Description	Type	Read Only	Sortable
id	system internal identifier for the audit	integer	Y	Y
status	status of the audit	enum(Audit Status)	Y	N
scanProfile	object reference to the scan profile used to perform the audit	objref	Y	Y
network	object reference to the network scanned in the audit	objref	Y	Y
dp	object reference to the Device Profiler used to perform the audit	objref	Y	Y
startTime	time at which the audit was started	integer	Y	Y
endTime	time at which the audit was completed or cancelled	integer	Y	Y
scanType	type of scan (for example, scheduled)	enum(ScanType)	Y	N
ASPLVersion	version of the ASPL database used for the scan	string	Y	N

Instance Methods

getReport

Generate an XML or CSV format report for the audit. The returned HTTP path should be queried to retrieve the report.
Minimum required access rights = Read-Only or Read-Scan on the network
Returns = an HTTP path to the report (string)

Table 6. "getReport" Arguments

Name	Description	Type	Optional
format	report format (valid values are "XML3", "XML2", "XML", "CSV2", and "CSV")	string	N
language	determines the language for the report 1 = English (default) 2 = Japanese	integer	Y

Class Methods

None.

Class: Credential

This abstract class represents credentials in IP360. This class defines attributes and methods common to all Credentials, but the system will never return a reference to an instance of this class: An SMB credential will be represented by a reference to an instance of the SMB class, an SNMP credential by a reference to an instance of the SNMP class, and so on.

Attributes

Table 7. "Credential" Class Attributes

Name	Description	Type	Read Only	Sortable
id	system internal identifier for the credential	integer	Y	Y
customer	object reference to the customer to whom this credential is assigned	objref	Y	Y
type	the type of credential	enum(Cred Type)	Y	N
name	the credential's name	string	N	N
notes	optional notes for the credential	string	N	N

Instance Methods

None.

Class Methods

None.

Class: CredSMB

This class represents SMB credentials. See "Class: Credential" on page 20 for more information about the abstract credential class.

Attributes

Table 8. "CredSMB" Class Attributes

Name	Description	Type	Read Only	Sortable
customer	object reference to the customer to whom this credential is assigned	objref	Y	Y
userName	SMB username to use with this credential	string	N	N
domain	domain to which this credential applies	string	N	N
authAlgorithm	minimum authentication algorithm to use for this credential	enum(CredSMBLevel)	N	N

Class Methods

create

Create a new SMB credential.
Minimum required access rights = Read-Write for Credentials
Returns = object reference to the new credential (objref)

Table 9. "create" Arguments

Name	Description	Type	Optional
name	new credential's name	string	N
username	credential's username	string	N
password	credential's password	string	N
authalgo	minimum authentication algorithm to use in this credential. This argument's value can be found in the enumType "CredSMBLevel."	integer	N
domain	domain to which this credential applies	string	Y
customer	object reference of the customer to whom this credential belongs	objref	Y

Instance Methods

setPassword

Modify an existing SMB credential's password.
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 10. "setPassword" Arguments

Name	Description	Type	Optional
password	credential's new password	string	N

delete

Deletes the credential
Minimum required access rights = Read-Write for Credentials
Returns = void
No arguments.

bindtoNetwork

Binds the credential to the specified network
Minimum required access rights = Read-Write for Credentials, Read-Only for Networks
Returns = void

Table 11. "bindToNetwork" Arguments

Name	Description	Type	Optional
network	object reference of the network to which to bind the credential	objref	N
addrs	IP block in that network to which to restrict the credential. The string values can be separated by commas ['10.10.140.0','10.10.140.1'], can contain a range ['10.10.140.0-10.10.140.3'], and can contain CIDR notation ['10.10.140.0/24'].	list(string)	Y

deleteBinding

Undoes the binding between a credential and a network
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 12. "deleteBinding" Arguments

Name	Description	Type	Optional
network	object reference of the network bound to the credential that should be unbound	objref	N

listBinding

Lists the networks and (optional) IP blocks to which the credential is bound
Minimum required access rights = Read-Only for Credentials
Returns = mixed (possible network objrefs and IP block list(string))
No arguments.

Class: CredSSHPassword

This class represents SSH password credentials. See "Class: Credential" on page 20 for more information about the abstract credential class.

Attributes

Table 13. "CredSSHPassword" Class Attributes

Name	Description	Type	Read Only	Sortable
customer	object reference to the customer to whom this credential belongs	objref	Y	Y
userName	SSH username to use with this credential	string	N	N

Class Methods

create

Create a new SSH password credential.
Minimum required access rights = Read-Write for Credentials
Returns = object reference to the new credential (objref)

Table 14. "create" Arguments

Name	Description	Type	Optional
name	new credential's name	string	N
username	credential's username	string	N
password	credential's password	string	N
customer	object reference of the customer to whom this credential belongs	objref	Y
notes	notes to associate with this credential	string	Y

Instance Methods

setPassword

Modify an existing SSH password credential's password.
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 15. "setPassword" Arguments

Name	Description	Type	Optional
password	credential's new password	string	N

delete

Deletes the credential
Minimum required access rights = Read-Write for Credentials
Returns = void
No arguments.

bindtoNetwork

Binds the credential to the specified network
Minimum required access rights = Read-Write for Credentials, Read-Only for Networks
Returns = void

Table 16. "bindToNetwork" Arguments

Name	Description	Type	Optional
network	object reference of the network to which to bind the credential	objref	N
addrs	IP block in that network to which to restrict the credential. The string values can be separated by commas ['10.10.140.0','10.10.140.1'], can contain a range ['10.10.140.0-10.10.140.3'], and can contain CIDR notation ['10.10.140.0/24'].	list(string)	Y

deleteBinding

Undoes the binding between a credential and a network
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 17. "deleteBinding" Arguments

Name	Description	Type	Optional
network	object reference of the network bound to the credential that should be unbound	objref	N

listBinding

Lists the networks and (optional) IP blocks to which the credential is bound
Minimum required access rights = Read-Only for Credentials
Returns = mixed (possible network objrefs and IP block list(string))
No arguments.

Class: CredSSHKey

This class represents SSH key credentials. See "Class: Credential" on page 20 for more information about the abstract credential class.

Attributes

Table 18. "CredSSHKey" Class Attributes

Name	Description	Type	Read Only	Sortable
customer	object reference to the customer to whom this credential belongs	objref	Y	Y
userName	SSH username to use with this credential	string	N	N

Class Methods

create

Create a new SSH key credential.
Minimum required access rights = Read-Write for Credentials
Returns = object reference to the new credential (objref)

Table 19. "create" Arguments

Name	Description	Type	Optional
name	new credential's name	string	N
username	credential's username	string	N
key	credential's key	string	N
customer	object reference of the customer to whom this credential belongs	objref	Y
notes	notes to associate with this credential	string	Y

Instance Methods

setKey

Modify an existing SSH key credential's key.
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 20. "setKey" Arguments

Name	Description	Type	Optional
key	credential's new key	string	N

delete

Deletes the credential
Minimum required access rights = Read-Write for Credentials
Returns = void
No arguments.

bindtoNetwork

Binds the credential to the specified network
Minimum required access rights = Read-Write for Credentials, Read-Only for Networks
Returns = void

Table 21. "bindToNetwork" Arguments

Name	Description	Type	Optional
network	object reference of the network to which to bind the credential	objref	N
addrs	IP block in that network to which to restrict the credential. The string values can be separated by commas ['10.10.140.0','10.10.140.1'], can contain a range ['10.10.140.0-10.10.140.3'], and can contain CIDR notation ['10.10.140.0/24'].	list(string)	Y

deleteBinding

Undoes the binding between a credential and a network
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 22. "deleteBinding" Arguments

Name	Description	Type	Optional
network	object reference of the network bound to the credential that should be unbound	objref	N

listBinding

Lists the networks and (optional) IP blocks to which the credential is bound
Minimum required access rights = Read-Only for Credentials
Returns = mixed (possible network objrefs and IP block list(string))
No arguments.

Class: CredSNMP

This class represents SNMP credentials. See "Class: Credential" on page 20 for more information about the abstract credential class.

Attributes

Table 23. "CredSNMP" Class Attributes

Name	Description	Type	Read Only	Sortable
customer	object reference to the customer to whom this credential belongs	objref	Y	Y

Class Methods

create

Create a new SNMP credential.
Minimum required access rights = Read-Write for Credentials
Returns = object reference to the new credential (objref)

Table 24. "create" Arguments

Name	Description	Type	Optional
name	new credential's name	string	N
commstr	community string to be used to authenticate	string	N
customer	object reference of the customer to whom this credential belongs	objref	Y
notes	notes to associate with this credential	string	Y

Instance Methods

setCommString

Modify an existing SNMP's credential's community string.
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 25. "setCommString" Arguments

Name	Description	Type	Optional
CommStr	credential's new community string	string	N

delete

Deletes the credential
Minimum required access rights = Read-Write for Credentials
Returns = void
No arguments.

bindtoNetwork

Binds the credential to the specified network
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 26. "bindToNetwork" Arguments

Name	Description	Type	Optional
network	object reference of the network to which to bind the credential	objref	N
addrs	IP block in that network to which to restrict the credential. The string values can be separated by commas ['10.10.140.0','10.10.140.1'], can contain a range ['10.10.140.0-10.10.140.3'], and can contain CIDR notation ['10.10.140.0/24'].	list(string)	Y

deleteBinding

Undoes the binding between a credential and a network
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 27. "deleteBinding" Arguments

Name	Description	Type	Optional
network	object reference of the network bound to the credential that should be unbound	objref	N

listBinding

Lists the networks and (optional) IP blocks to which the credential is bound
Minimum required access rights = Read-Only for Credentials
Returns = mixed (possible network objrefs and IP block list(string))
No arguments.

Class: CredWebForm

This class represents credentials for web forms requiring authentication. See "Class: Credential" on page 20 for more information about the abstract credential class.

Attributes

Table 28. "CredWebForm" Class Attributes

Name	Description	Type	Read Only	Sortable
customer	object reference to the customer to whom this credential belongs	objref	Y	Y
userName	the username for the credential	string	N	N
loginUrl	login url to use with the credential	string	N	N

Class Methods

create

Create a new credential for web form authentication.
Minimum required access rights = Read-Write for Credentials
Returns = object reference to the new credential (objref)

Table 29. "create" Arguments

Name	Description	Type	Optional
name	new credential's name	string	N
username	the username for the credential	string	N
password	the password for the credential	string	N
loginurl	login url to use with the credential	string	N
customer	object reference of the customer to whom this credential belongs	objref	Y

Instance Methods

setPassword

Modify an existing password for a web form credential.
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 30. "setPassword" Arguments

Name	Description	Type	Optional
password	the password for the web form	string	N

delete

Deletes the credential
Minimum required access rights = Read-Write for Credentials
Returns = void
No arguments.

bindtoNetwork

Binds the credential to the specified network
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 31. "bindToNetwork" Arguments

Name	Description	Type	Optional
network	object reference of the network to which to bind the credential	objref	N
addrs	IP block in that network to which to restrict the credential. The string values can be separated by commas ['10.10.140.0','10.10.140.1'], can contain a range ['10.10.140.0-10.10.140.3'], and can contain CIDR notation ['10.10.140.0/24'].	list(string)	Y

deleteBinding

Undoes the binding between a credential and a network
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 32. "deleteBinding" Arguments

Name	Description	Type	Optional
network	object reference of the network bound to the credential that should be unbound	objref	N

listBinding

Lists the networks and (optional) IP blocks to which the credential is bound
Minimum required access rights = Read-Only for Credentials
Returns = mixed (possible network objrefs and IP block list(string))

No arguments.

bindToVirtualHost

Binds the credential to the specified virtual hosts.
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 33. "bindtoVirtualHost" Arguments

Name	Description	Type	Optional
network	object reference of the network containing the virtual host to which to bind the credential	objref	N
ipaddress	IP address of the virtual host	string	N
hostnames	hostname of the virtual host	string	N

deleteBindingToVirtualHost

Undoes the binding between a credential and a virtual host.
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 34. "deleteBindingToVirtualHost" Arguments

Name	Description	Type	Optional
network	object reference of the network containing the virtual host from which the credential will be deleted	objref	N
ipaddress	IP address of the virtual host	string	N

listBindingToVirtualHost

Lists the networks, IP addresses, and hostnames of the virtual hosts to which the credential is bound. An optional URL may also be listed.
Minimum required access rights = Read-Only for Credentials
Returns = mixed (possible network objrefs, IP address(string), and hostnames(string))

Table 35. "listBindingToVirtualHost" Arguments

Name	Description	Type	Optional
network	object reference of the network	objref	N

Class: CredWebHTTP

This class represents credentials for web applications requiring authentication. See "Class: Credential" on page 20 for more information about the abstract credential class.

Attributes

Table 36. "CredWebHTTP" Class Attributes

Name	Description	Type	Read Only	Sortable
customer	object reference to the customer to whom this credential belongs	objref	Y	Y
userName	the username for the credential	string	N	N
authMethod	authentication method for the credential 1 = Basic 2 = Digest 3 = NTLM 4 = determined at scan time; strongest method used	enum(CredWebLevel)	N	N

Class Methods

create

Create a new credential for authorizing web applications.
Minimum required access rights = Read-Write for Credentials
Returns = object reference to the new credential (objref)

Table 37. "create" Arguments

Name	Description	Type	Optional
name	new credential's name	string	N
username	the username for the credential	string	N
password	the password for the credential	string	N
authMethod	authentication method for the credential 1 = Basic 2 = Digest 3 = NTLM 4 = determined at scan time; strongest method used	integer	N
customer	object reference of the customer to whom this credential belongs	objref	Y

Instance Methods

setPassword

Modify an existing password.
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 38. "setPassword" Arguments

Name	Description	Type	Optional
password	the password for the credential	string	N

delete

Deletes the credential
Minimum required access rights = Read-Write for Credentials
Returns = void
No arguments.

bindtoNetwork

Binds the credential to the specified network
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 39. "bindToNetwork" Arguments

Name	Description	Type	Optional
network	object reference of the network to which to bind the credential	objref	N
addrs	IP block in that network to which to restrict the credential. The string values can be separated by commas ['10.10.140.0','10.10.140.1'], can contain a range ['10.10.140.0-10.10.140.3'], and can contain CIDR notation ['10.10.140.0/24'].	list(string)	Y

deleteBinding

Undoes the binding between a credential and a network
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 40. "deleteBinding" Arguments

Name	Description	Type	Optional
network	object reference of the network bound to the credential that should be unbound	objref	N

listBinding

Lists the networks and (optional) IP blocks to which the credential is bound
Minimum required access rights = Read-Only for Credentials
Returns = mixed (possible network objrefs and IP block list(string))

No arguments.

bindToVirtualHost

Binds the credential to the specified virtual hosts.
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 41. "bindtoVirtualHost" Arguments

Name	Description	Type	Optional
network	object reference of the network containing the virtual host to which to bind the credential	objref	N
ipaddress	IP address of the virtual host	string	N
hostnames	hostname of the virtual host	string	N

deleteBindingToVirtualHost

Undoes the binding between a credential and a virtual host.
Minimum required access rights = Read-Write for Credentials
Returns = void

Table 42. "deleteBindingToVirtualHost" Arguments

Name	Description	Type	Optional
network	object reference of the network containing the virtual host from which the credential will be deleted	objref	N
ipaddress	IP address of the virtual host	string	N

listBindingToVirtualHost

Lists the networks, IP addresses, and hostnames of the virtual hosts to which the credential is bound. An optional URL may also be listed.
Minimum required access rights = Read-Only for Credentials
Returns = mixed (possible network objrefs, IP address(string), and hostnames(string))

Table 43. "listBindingToVirtualHost" Arguments

Name	Description	Type	Optional
network	object reference of the network	objref	N

Class: Customer

This class represents customer records in the IP360 system. A typical IP360 installation has a single customer record, but some IP360 systems are configured to serve more than one customer. In the multiple-customer case, one customer is designated the "master customer." Administrator users of the master customer may create and modify all customer records. Administrator users of non-master customers may modify only their own customer record.

Attributes

Table 44. "Customer" Class Attributes

Name	Description	Type	Read Only	Sortable
id	system internal identifier for the customer	integer	Y	Y
name	name of the customer	string	Y	Y
mailserver	SMTP server for the customer (must be a valid IP address)	string	N	N
alertEmailAddress	email address for customer alerts	string	N	N
masterCustomer	true if the customer is the master customer	boolean	Y	N
redScoreThreshold	mandatory red score threshold for the customer's users	integer	Y	N
timezone	default timezone for the customer's users	string	N	N
language	default language for the customer's users	enum(language)	N	N
UITimeout	number of minutes before idle UI sessions are terminated	integer	N	N
deleted	true if customer is deleted	boolean	Y	N

Instance Methods

setRedScoreThreshold

Set the mandatory red score threshold for all users. If this parameter is set, no user for this customer may change the red score threshold.
Minimum required access rights = Read-Write on VnE Manager
Returns = void

Table 45. "setRedScoreThreshold" Arguments

Name	Description	Type	Optional
score	red score threshold value to be set	integer	N

clearRedScoreThreshold

Clear the mandatory red score threshold value. This permits users to change their red score thresholds.
Minimum required access rights = Read-Write on VnE Manager
Returns = void
No arguments.

setIP360Reporting

Sets reporting to internal IP360 reporting. This is the default way of running reports.
Minimum required access rights = Read-Write on VnE Manager
Returns = void
No arguments.

setSIHReporting

Sets reporting to the url of an external Suite360 Intelligence Hub server.
Minimum required access rights = Read-Write on VnE Manager
Returns = void

Table 46. "setSIHReporting" Arguments

Name	Description	Type	Optional
url	url of the Suite360 Intelligence Hub server; for example: `https://<hostname>/sih`	string	N

getExternalUrl

When reporting is set to external url, this method can be called to retrieve the url.
Minimum required access rights = Read-Only on VnE Manager
Returns = a string containing the external reporting url
No arguments.

setCustIPSpace

Add an IPspace to the customer
Minimum required access rights = Read-Write on VnE Manager
Returns = void
No arguments.

Table 47. "setCustIPSpace" Arguments

Name	Description	Type	Optional
range	Range of IP addresses for the customer IP Space. The range must be a list of strings, where each string contains an IP address in dotted-quad form (for example, 10.1.1.5), a CIDR block (for exmaple, 10.1.1.0/24), or an IP address range (for example, 10.3.40.0-10.3.44.10).	string	N

getCustIPSpace

Returns a list of the customer IP space
Minimum required access rights = Read-Only on VnE Manager
Returns = ranges of IP addresses in the customer IP Space
No arguments.

Class Methods

None.

Class: DP

This class represents the Device Profiler appliances connected to the IP360 system. See "Class: Appliance" on page 16 for more information about the appliance abstraction class.

Please note that after the `startScan` method is invoked, the API immediately returns an object reference to the scan. However, the actual scan object does not exist until the scan starts, which can be a while after the method is invoked, depending on the DP's queue of scans to be performed. In the interim, any method invoked on that scan object reference will result in an "Invalid object reference" error.

Attributes

Table 48. "DP" Class Attributes

Name	Description	Type	Read Only	Sortable
customer	object reference to the customer to whom this appliance is assigned	objref	Y	Y
softConnectionLimit	customer-configured connection limit for the DP	integer	Y	N
hardConnectionLimit	override connection limit for the DP	integer	Y	N
hardwareConnectionLimit	connection limit for the DP, as determined by the appliance hardware	integer	Y	N
lastHeartbeat	time the DP last checked in with the VnE	integer	Y	N

Instance Methods

setSoftConnectionLimit

Change the soft connection limit for the Device Profiler.
Minimum required access rights = Read-Write on the Device Profiler
Returns = void

Table 49. "setSoftConnectionLimit" Arguments

Name	Description	Type	Optional
limit	new connection limit	integer	N

getActiveScans

Get a list of active (status = "In Progress, Paused, Auto-paused, Suspended") audits for the Device Profiler.
Minimum required access rights = Read-Only on the Device Profiler
Returns = a list or table (list of lists) of the "In Progress, Paused, Auto-paused, Suspended" scans performed by the DP (searchResult)

Table 50. "getActiveScans" Arguments

Name	Description	Type	Optional
format	valid values are "list" (to return a list of object references) and "table" (to return a table of object attributes). Defaults to "list." See "Search Result Formats" on page 9 for a description of how search results are formatted.	string	Y

getASPLVersion

Get the ASPL version for the Device Profiler.
Minimum required access rights = Read-Only on the Device Profiler
Returns = the ASPL version

startScan

*Scan a network using this Device Profiler, using a specified scan profile.**
Minimum required access rights = Read-Scan on the network, Read-Write on the Device Profiler, and Read-Only on the scan profile
Returns = object reference to the audit (scan) that contains the scan results (objref)

Table 51. "startScan" Arguments

Name	Description	Type	Optional
network	object reference to the network to scan	objref	N
scanProfile	object reference to the scan profile to use	objref	N
range	the subset of the network to scan. The range must be a list of strings, where each string contains an IP address in dotted-quad form (for example, 10.1.1.5), a CIDR block (for exmaple, 10.1.1.0/24), or an IP address range (for example, 10.3.40.0-10.3.44.10).	list (string)	Y

* As is the case with scans manually started in the IP360 UI, manually starting scans will pause all scheduled scans currently in progress on the Device Profiler until all manually started scans have completed. See "pauseScan" on page 39 for more information.

cancelScan

Cancel an audit currently in progress on this Device Profiler.
Minimum required access rights = Read-Write on the network, Read-Only on the Device Profiler, and Read-Only on the scan profile
Returns = void

Table 52. "cancelScan" Arguments

Name	Description	Type	Optional
audit	object reference to the audit (scan) to cancel; to cancel suspended scans, use scan state = 7.	objref	N

pauseScan

Pause an audit currently in progress on this Device Profiler. The audit may be resumed using the `resumeScan` *method.*
Minimum required access rights = Read-Write on the network and Read-Only on the DP

Returns = void

Table 53. "pauseScan" Arguments

Name	Description	Type	Optional
audit	object reference to the audit (scan) to pause	objref	N

resumeScan

Resume a paused audit on this Device Profiler.
Minimum required access rights = Read-Write on the network and Read-Only on the DP
Returns = void

Table 54. "resumeScan" Arguments

Name	Description	Type	Optional
audit	object reference to the audit (scan) to resume	objref	N

Class Methods

None.

Class: GlobalIpExclude

This class represents the "Global IP Exclude" list, which instructs IP360 not to scan the specified IP addresses.

Attributes

Table 55. "GlobalIp Exclude" Class Attributes

Name	Description	Type	Read Only	Sortable
id	system internal identifier for the exclude list	integer	Y	Y
customer	object reference to the customer to whom the exclude list belongs	objref	Y	Y
ipaddress	list of IP addresses included in the exclude list (that is, to be excluded from scanning)	string	Y	N
notes	notes to associate with the exclude list	string	N	N

Class Methods

create

Creates a list of globally excluded IP addresses.
Minimum required access rights = Read-Write access to Globally Excluded IPs
Returns = object reference to the new object

Table 56. "create" Arguments

Name	Description	Type	Optional
addrs	IP block in that network to which to restrict the credential. The string values can be separated by commas ['10.10.140.0','10.10.140.1'], can contain a range ['10.10.140.0-10.10.140.3'], and can contain CIDR notation ['10.10.140.0/24'].	list(string)	N
customer	object reference to the customer to whom the exclude list belongs	objref	Y
notes	notes to associate with the exclude list	string	Y

list

Returns a list of globally excluded IP addresses.
Minimum required access rights = Read-Only access to Globally Excluded IPs
Returns = list (global exclude list ID, [list of excluded IPs, blocks, and ranges])

Table 57. "list" Arguments

Name	Description	Type	Optional
customer	object reference to the customer whose Global Exclude IP lists will be returned. If not specified, the user's customer is used.	objref	Y

Instance Methods

delete

Deletes the object for a list of globally excluded IP addresses.
Minimum required access rights = Read-Write on the network
Returns = void
No arguments.

appendExcludeIps

Appends to the ip address space of a globally excluded IP address.
Minimum required access rights = Read-Write access to Globally Excluded IPs

Returns = void

Table 58. "appendExcludeIps" Arguments

Name	Description	Type	Optional
addrs	addresses to append to the list of Globally Excluded IPs	list (string)	N

updateExcludeIps

Updates ip address space of a globally excluded IP address.
Minimum required access rights = Read-Write access to Globally Excluded IPs
Returns = void

Table 59. "updateExcludeIps" Arguments

Name	Description	Type	Optional
addrs	addresses to add to Globally Excluded IPs	list (string)	N

Class: IPSnTellect

This class represents the nTellect appliances connected to the IP360 system. See "Class: Appliance" on page 16 for more information about the appliance abstraction class.

Attributes

Table 60. "IPSnTellect" Class Attributes

Name	Description	Type	Read Only	Sortable
customer	object reference to the customer to whom this appliance is assigned	objref	Y	Y
lastHeartbeat	time the nTellect last checked in with the VnE Manager	integer	Y	N

Instance Methods

None.

Class Methods

None.

Class: Network

This class represents the networks configured in the IP360 system. A network is a collection of IP space that can be specified as a scan target for a Device Profiler or a monitoring target for an IPS nTellect.

Attributes

Table 61. "Network" Class Attributes

Name	Description	Type	Read Only	Sortable
id	system internal identifier the network	integer	Y	Y
customer	object reference to the customer to which this network belongs	objref	Y	Y
name	name of the network	string	N	Y
active	true if the network is active	boolean	N	N
static	true if the network has static IP address assignments	boolean	Y	N
assetValue	asset value of the network	integer	Y	N
notes	notes for the network	string	N	N
NetBIOSPrimary	true if reports should show the NetBIOS hostname for hosts in this network instead of the DNS name	boolean	N	N
include	list of IP addresses included in the network	list(string)	Y	N
exclude	list of IP addresses excluded from the network	list(string)	Y	N
deleted	true if the network is deleted	boolean	Y	N

Instance Methods

delete

Delete the network.
Minimum required access rights = Read-Write on the network
Returns = void
No arguments.

enableDHT

Enables dynamic host tracking on the specified network.
Minimum required access rights = Read-Write on the network
Returns = void

Table 62. "enableDHT" Arguments

Name	Description	Type	Optional
correlation	specifies the name resolution correlation Values are: 2 = NetBIOS 3 = DNS	integer	N

disableDHT

Disables dynamic host tracking on the specified network.
Minimum required access rights = Read-Write on the network
Returns = void
No arguments.

getOwner

Return the owner of the network, or the empty string if the network has no owner..
Minimum required access rights = Read-Only on the network
Returns = object reference to the user or user group that is the network owner (objref)
No arguments.

setOwner

Change the owner of the network.
Minimum required access rights = Read-Write on the network
Returns = void

Table 63. "setOwner" Arguments

Name	Description	Type	Optional
owner	object reference to the user or user group that is the new owner	objref	N

clearOwner

Clear the owner of the network.
Minimum required access rights = Read-Write on the network

Returns = void
No arguments

setAssetValue

Change the asset value of the network.
Minimum required access rights = Read-Write on the network
Returns = void

Table 64. "setAssetValue" Arguments

Name	Description	Type	Optional
value	new asset value (valid values are between 1 and 999999999)	integer	N

clearAssetValue

Clear the asset value of the network.
Minimum required access rights = Read-Write on the network
Returns = void
No arguments.

addIncludes

Add one or more IP addresses, IP address ranges, or CIDR blocks to the set of addresses included in the network definition.
Minimum required access rights = Read-Write on the network
Returns = void

Table 65. "addIncludes" Arguments

Name	Description	Type	Optional
addrs	Addresses to add to the network's "include" list	list (string)	N

deleteIncludes

Remove one or more IP addresses, IP address ranges, or CIDR blocks from the set of addresses included in the network definition.
Minimum required access rights = Read-Write on the network
Returns = void

Table 66. "deleteIncludes" Arguments

Name	Description	Type	Optional
addrs	Addresses to remove from the network's "exclude" list	list (string)	N

addExcludes

Add one or more IP addresses, IP address ranges, or CIDR blocks to the set of addresses excluded from the network definition.
Minimum required access rights = Read-Write on the network

Returns = void

Table 67. "addExcludes" Arguments

Name	Description	Type	Optional
addrs	Addresses to add to the network's "exclude" list	list (string)	N

deleteExcludes

Remove one or more IP addresses, IP address ranges, or CIDR blocks from the set of addresses excluded from the network definition.
Minimum required access rights = Read-Write on the network
Returns = void

Table 68. "deleteExcludes" Arguments

Name	Description	Type	Optional
addrs	Addresses to remove from the network's "exclude" list	list (string)	N

The following instance methods all pertain to Virtual Hosts:

getVirtualHosts

Returns a list of virtual host defined on the network.
Minimum required access rights = Read-Only on the network
Returns = List of virtual host records, each of which consists of an IP address or range and a list of FQDN hostnames (list (string, list (string)))
No arguments.

addVirtualHost

Defines a new virtual host on a network.
Minimum required access rights = Read-Write on the network
Returns = Void.

Table 69. "addVirtualHost" Arguments

Name	Description	Type	Optional
address	IP address or range (for example, 10.10.10.1 or 10.10.10.1–10.10.10.5) of the virtual host	string	N
hostnames	Hostnames (FQDNs) to associate with the virtual host	list (string)	N

Table 69. "addVirtualHost" Arguments

Name	Description	Type	Optional
pathinclude	Included paths for scanning; refer to the *Scanning Domain-Based Virtual Hosts* topic in the *IP360 Administration Guide* for format and excluded characters.	list (string)	Y
pathexclude	Excluded paths for scanning; refer to the *Scanning Domain-Based Virtual Hosts* topic in the *IP360 Administration Guide* for format and excluded characters.	list (string)	Y

deleteVirtualHost

Deletes an existing virtual host from a network.
Minimum required access rights = Read-Write on the network
Returns = Void

Table 70. "deleteVirtualHost" Arguments

Name	Description	Type	Optional
address	IP address or range (for example, 10.10.10.1 or 10.10.10.1–10.10.10.5) of the virtual host. This must exactly match the address originally used in addVirtualHost to create the virtual host.	string	N

Class Methods

create

Create a new network.
Minimum required access rights = Read-Write on Networks
Returns = object reference to the new network instance (objref)

Table 71. "create" Arguments

Name	Description	Type	Optional
name	new network's name	string	N
customer	object reference to the customer for whom to create network	objref	Y

Class: NetworkGroup

This class represents the network groups configured in the IP360 system. A network group is a collection of networks and network groups organized for common reporting and administration.

Attributes

Table 72. "NetworkGroup" Class Attributes

Name	Description	Type	Read Only	Sortable
id	system internal identifier for the network group	integer	Y	Y
customer	object reference to the customer to whom this network group belongs	objref	Y	Y
name	the network group's name	string	N	Y
parentGroup	object reference to the parent group of this network group	objref	Y	Y
assetValue	the network group's asset value	integer	Y	N
notes	notes for the network group	string	N	N

Instance Methods

delete

Delete the network group.
Minimum required access rights = Read-Write on the network group
Returns = void
No arguments.

getOwner

Return the owner of the network group, or the empty string if the network group has no owner.
Minimum required access rights = Read-Only on the network group
Returns = object reference to the user or user group that is the network group owner (objref)
No arguments.

setOwner

Change the owner of the network group.
Minimum required access rights = Read-Write on the network group
Returns = void

Table 73. "setOwner" Arguments

Name	Description	Type	Optional
owner	object reference to the user or user group that is new owner	objref	N

clearOwner

Clear the owner of the network group.
Minimum required access rights = Read-Write on the network group

Returns = void
No arguments.

setAssetValue

Change the asset value of the network group.
Minimum required access rights = Read-Write on the network group
Returns = void

Table 74. "setAssetValue" Arguments

Name	Description	Type	Optional
value	new asset value (valid values are between 1 and 999999999)	integer	N

clearAssetValue

Clear the asset value of the network group.
Minimum required access rights = Read-Write on the network group
Returns = void
No arguments

setParentGroup

Set the parent group of the network group.
Minimum required access rights = Read-Write on the network group and the parent network group
Returns = void

Table 75. "setParentGroup" Arguments

Name	Description	Type	Optional
parent	object reference to the parent network group	objref	N

clearParentGroup

Unset the parent group of the network group.
Minimum required access rights = Read-Write on the network group and the parent network group
Returns = void
No arguments.

getMembers

Return a list of the networks in the group.
Minimum required access rights = Read-Only on the network group

Returns = a list or table (list of lists) of the network in the network group (searchResult)

Table 76. "getMembers" Arguments

Name	Description	Type	Optional
format	Values are "list" (to return a list of object references) and "table" (to return a table of object attributes). Defaults to "list." See "Search Result Formats" on page 9 for a description of how search results are formatted.	string	Y

addMember

Add a network to the network group.
Minimum required access rights = Read-Write on the network group and on the network
Returns = void

Table 77. "addMember" Arguments

Name	Description	Type	Optional
network	object reference to the network to add to the group	objref	N

removeMember

Remove a network from the network group.
Minimum required access rights = Read-Write on the network and the network group
Returns = void

Table 78. "removeMember" Arguments

Name	Description	Type	Optional
network	object reference to the network to remove from the network group	objref	N

Class Methods

create

Create a new network group.
Minimum required access rights = Read-Write on Network Groups

Returns = object reference to the new network group instance (objref)

Table 79. "create" Arguments

Name	Description	Type	Optional
name	new network group's name	string	N
parentGroup	object reference to the parent group of new network group	objref	Y
customer	object reference to the customer for whom to create network group	objref	Y

Class: Role

This class represents roles in the IP360 system.

Attributes

Table 80. "Role" Class Attributes

Name	Description	Type	Read Only	Sortable
id	system internal identifier for the role	integer	Y	N
customer	object reference to the customer to which this role belongs	objref	Y	N
name	name of the role	string	N	N

Instance Methods

delete

Delete the role.
Minimum required access rights = Read-Write on the role
Returns = void
No arguments.

getUsers

Get a list of users assigned to this role.
Minimum required access rights = Read-Only on the role

Returns = a list or table (list of lists) of the role's users (searchResult)

Table 81. "getUsers" Arguments

Name	Description	Type	Optional
format	Values are "list" (to return a list of object references) and "table" (to return a table of object attributes). Defaults to "list." See "Search Result Formats" on page 9 for a description of how search results are formatted.	string	Y

Class Methods

None.

Class: ScanConfiguration

This class represents the scan configurations configured in the IP360 system. A scan configuration specifies a network to scan, a Device Profiler to perform the scan, and a scan profile to instruct the Device Profiler how to perform the scan.

Attributes

Table 82. "ScanConfiguration" Class Attributes

Name	Description	Type	Read Only	Sortable
id	system internal identifier for the scan configuration	integer	Y	Y
network	object reference to the network scanned for this scan configuration	objref	Y	Y
scanProfile	object reference to the scan profile used by this scan configuration	objref	Y	Y
dp	object reference to the Device Profiler that performs the scan for this scan configuration	objref	Y	Y
active	true if the scan configuration is active	boolean	N	N

Instance Methods

delete

Delete the scan configuration.
Minimum required access rights = Read-Write
Returns = void
No arguments.

Class Methods

create

Create a new scan configuration.
Minimum required access rights = Read-Write
Returns = object reference to the new scan configuration instance (objref)

Table 83. "create" Arguments

Name	Description	Type	Optional
network	object reference to the network in the scan configuration	objref	N
scanProfile	object reference to the scan profile in the scan configuration	objref	N
dp	object reference to the Device Profiler in the scan configuration	objref	N

Class: ScanProfile

This class represents the scan profiles configured in the IP360 system. A scan profile is a set of instructions to the Device Profiler specifying which scan operations to perform during a scan of a network.

Attributes

Table 84. "ScanProfile" Class Attributes

Name	Description	Type	Read Only	Sortable
id	system internal identifier for the scan profile	integer	Y	Y
customer	object reference to the customer to whom this scan profile belongs	objref	Y	Y

Table 84. "ScanProfile" Class Attributes

Name	Description	Type	Read Only	Sortable
name	user-specified name of the scan profile	string	N	Y
profile	XML code that defines the scan profile; default XML file (*ip360_scan_profile.xml*) can be exported from the Scan Profile screen	string	Y	N
active	true if the scan profile is active	boolean	N	N
deleted	true if the scan profile is deleted	boolean	Y	N

Instance Methods

setProfile

Modify the scan profile.
Minimum required access rights = Read-Write on Scan Profiles
Returns = void

Table 85. "setProfile" Arguments

Name	Description	Type	Optional
profile	XML code that (re)defines the profile; default XML file (*ip360_scan_profile.xml*) can be exported from the Scan Profile screen	string	N

delete

Delete the scan profile.
Minimum required access rights = Read-Write on Scan Profiles
Returns = void
No arguments.

Class Methods

create

Create a new scan profile.
Minimum required access rights = Read-Write on Scan Profiles

Returns = object reference to the new scan profile (objref)

Table 86. "create" Arguments

Name	Description	Type	Optional
name	scan profile's name	string	N
profile	XML code that defines the scan profile; default XML file (*ip360_scan_profile.xml*) can be exported from the Scan Profile screen	string	N
active	whether the scan profile is active (enabled) or inactive (disabled)	boolean	N
customer	object reference to the customer to whom the scan profile belongs	objref	Y

Class: ThreatZoneGroup

This class represents Topology Threat Zone Groups. Threat Zone Groups are used in Topological Risk Analysis to define perspectives from which to evaluate line of sight to network hosts and the expected threat level from each of those perspectives. Threat Zone Groups are bound to a Device Profiler.

For more information on Threat Zone Groups, refer to the *Topological Risk Analysis* chapter in the *IP360 Administrator's Guide*.

Attributes

Table 87. "ThreatZoneGroup" Class Attributes

Name	Description	Type	Read Only	Sortable
id	system internal identifier for the threat zone group	integer	Y	Y
customer	object reference to the customer to whom this threat zone group belongs	objref	Y	Y
name	user-specified name of the threat zone group	string	N	N
notes	description of the threat zone group	string	N	N
threatLevel	threat level to set for this threat zone group	enum	N	N

Instance Methods

delete

Delete this threat zone group.
Minimum required access rights = Read-Write on VnE Manager
Returns = void
No arguments.

listAvailableThreatZones

Generate a list of threat zones that are available to be added to a threat zone group.
Minimum required access rights = Read Only on VnE Manager

Returns = lists of Threat Zone Groups, with reference to Device Profiler and DP perspective. Following is an example of the return value:

```
['DP.1', 'dp17-vm.test.ncircle.com', '10.10.129.117/1']
```

In this example, `dp17-vm.test.ncircle.com` is the Device Profiler. `10.10.129.117/1` (DP IP Address/Prefix) is the perspective.

No arguments.

addThreatZone

Add a new threat zone to a threat zone group.
Minimum required access rights = Read-Write on VnE Manager
Returns = void

Table 88. "addThreatZone" Arguments

Name	Description	Type	Optional
name	name of the threat zone	string	N
perspective	DP perspective is retrieved through a call to `listAvaliableThreatZones;` DP perspective is DP's `IP Address/Prefix`	string	N
comment	optional descriptive comment	string	Y

deleteThreatZone

Delete a threat zone from a specific threat zone group.
Minimum required access rights = Read-Write on VnE Manager
Returns = void

Table 89. "deleteThreatZone" Arguments

Name	Description	Type	Optional
name	the name of a threat zone to be deleted	string	N

listThreatZones

List the threat zones in a threat zone group.
Minimum required access rights = Read Only on VnE Manager
Returns = list of Threat Zone Groups
No arguments.

Class Methods

create

Create a new threat zone group.
Minimum required access rights = Read-Write on VnE Manager
Returns = object reference to the new threat zone group (objref) or exception if unsuccessful

Table 90. "create" Arguments

Name	Description	Type	Optional
name	name of threat zone	string	N
threatLevel	integer from 1 - 5 that indicates level of threat for the threat zone group 1=Internet 2=Critical 3=High 4=Medium 5=Low	integer	N
customer	object reference to the customer to whom this threat zone group belongs	objref	Y

Class: TM

This class represents the Threat Monitor appliances connected to the IP360 system. (This class is applicable only if a Threat Monitor is part of the IP360 deployment.) See "Class: Appliance" on page 16 for more information about the appliance abstraction class.

Attributes

Table 91. "TM" Class Attributes

Name	Description	Type	Read Only	Sortable
customer	object reference to the customer to whom this appliance is assigned	objref	Y	Y
ignoreDP	true if the TM should ignore packets originating from Device Profilers	boolean	N	N
lastHeartbeat	time the TM last checked in with the VnE	integer	Y	N

Instance Methods

None.

Class Methods

None.

Class: TopoAnalysis

This class generates a Topological Risk Analysis (TRA) report. This class is almost identical to the Audit class, except it operates on many objects. See "Class: Audit" on page 18.

Attributes

Table 92. "TopoAnalysis" Class Attributes

Name	Description	Type	Read Only	Sortable
id	system internal identifier for topological analyis records	integer	Y	Y
network	object reference for the network for which the topological analysis was obtained	objref	Y	N
startTime	time at which the analysis was started	integer	Y	Y
endTime	time at which the analysis was completed	integer	Y	Y
status	status of the analysis	enum	Y	N
type	type of the analysis (Router or TTL)	enum	Y	N

Note: The startTime and endTime values are in epoch time integer format, which is calculated as time since Jan. 1, 1970. For example, if you have a startTime of Jan. 1, 2009 at 1:00 GMT that would be returned to you as 1230771600. See "Data Types" on page 16 for information about epoch time.

Instance Methods

getReport

Generates a special Topological Risk Analysis in XML3-type format. The report is in English. The report is retrieved from the returned HTTP path. Following is an example of the report:

```
<?xml version="1.0"?>
<analysis id="11" vne_name="IP360 VnE" vne_hostname="vm-vne14.test.ncircle.com">
  <status>Finished</status>
  <startDate>2008-08-21T20:42:41.47345+00:00</startDate>
  <endDate>2008-08-21T20:44:32.426205+00:00</endDate>
  <customer id="1">nCircle Network Security</customer>

  <network id="1">
    <name>one host</name>
    <ranges>
      <range type="include">10.10.130.21</range>
    </ranges>
  </network>

  <hosts>
    <host id="11" persistent_id="4" worth="55" hostthreat="0.78" >
      <ip>10.10.130.21</ip>
      <dnsName>10.10.130.21</dnsName>
      <netbiosName>AD-SUS</netbiosName>
      <netbiosDomainName>TEST-WIN2KAD</netbiosDomainName>
      <macAddress>00:03:47:7C:B8:88</macAddress>
      <score>2114</score>
      <os id="42"/>
      <applications>
        <application application_id="30" service_id="0" port="80" protocol="tcp"
appthreat="0.6">
          <vulnerabilities>
            <vulnerability id="60" custom="0" score="50"/>
          </vulnerabilities>
          <insecurityzones>
            <insecurityzone id="1" groupname="dmz" threat="0.75"/>
            <insecurityzone id="2" groupname="corp" threat="0.25"/>
            <insecurityzone id="3" groupname="lab" threat="0.5"/>
          </insecurityzones>
        </application>
        <application application_id="0" service_id="0" port="443" protocol="tcp"
appthreat="0.56">
          <vulnerabilities>
            <vulnerability id="6140" custom="0" score="0"/>
            <vulnerability id="4143" custom="0" score="1000"/>
          </vulnerabilities>
          <insecurityzones>
            <insecurityzone id="1" groupname="dmz" threat="0.75"/>
            <insecurityzone id="3" groupname="lab" threat="0.5"/>
          </insecurityzones>
        </application>
      </applications>
    </host>
  </hosts>
</analysis>
```

Minimum required access rights = Read-Only on the network
Returns = an HTTP path to the report (string)

Class Methods

None.

Class: TopoConfiguration

This class represents the configuration necessary to generate a Topological Risk Analysis (TRA) report.

Attributes

none

Instance Methods

none

Class Methods

setDisabled

Disables the Topology Risk Analyzer.
Minimum required access rights = none required
Returns = void

setEnabled

Enables the Topology Risk Analyzer.
Minimum required access rights = none required
Returns = void

setStartTime

Sets the date and time, in epoch time format, at which Topology Risk Analyzer will start being run.
Minimum required access rights = none required
Returns = void

Table 93. "setStartTime" Arguments

Name	Description	Type	Optional
starttime	date and time, in epoch time format, at which the Topology Risk Analyzer will start	string	N

setStopTime

Sets the date and time, in epoch time format, after which Topology Risk Analyzer should no longer be run. The stoptime is generally set significantly after the expiration of the subscription term.
Minimum required access rights = none required
Returns = void

Table 94. "setStopTime" Arguments

Name	Description	Type	Optional
stoptime	date and time, in epoch time format, for when the topology should no longer be run	string	N

Note: The starttime and stoptime values are in epoch time integer format, which is calculated as time since Jan. 1, 1970. For example, if you have a startTime of Jan. 1, 2009 at 1:00 GMT that would be returned to you as 1230771600. See "Data Types" on page 16 for information about epoch time.

setTopoFrequency

Sets the frequency for the topology.
Minimum required access rights = none required
Returns = void

Table 95. "setTopoFrequency" Arguments

Name	Description	Type	Optional
count	number of <unitoftime> between the topology analysis	integer	N
unitoftime	Sets the unit of time for the frequency of the topology. Values are: • minute • hour • day • week • month • year	string	N

Class: User

This class represents the users in the IP360 system.

Attributes

Table 96. "User" Class Attributes

Name	Description	Type	Read Only	Sortable
id	system internal identifier for the user	integer	Y	N
customer	object reference for the customer this user belongs to	objref	Y	N
firstName	user's first name	string	N	N
middleInitial	user's middle initial	string	N	N
lastName	user's last name	string	N	N
title	user's title	string	N	N
emailAddress	user's email address (this is also the user's login name)	string	N	Y
alertEmailAddress	email address to which alerts are sent for this user	string	N	N
skin	web UI skin for this user	enum(Skin)	N	N
lastLogin	time this user last logged in	integer	Y	N
unsuccessfulLogins	number of unsuccessful login attempts since the user's last successful login	integer	Y	N
disabledTime	if non-zero, the time at which the user's account was disabled because of unsuccessful login attempts	integer	Y	N
accountDisabled	if true, the user's account is administratively disabled	boolean	Y	N
passwordLastChanged	time at which the user's password was last changed	integer	Y	N
timezone	user's timezone preference	string	N	N
language	user's language preference	enum(Language)	N	N
phoneNumber	user's phone number	string	N	N
manager	name of the user's manager	string	N	N
deleted	true if the user is deleted	boolean	Y	N

Instance Methods

delete

Delete the user.
Minimum required access rights = Read-Write on the user
Returns = void
No arguments.

setPassword

Set the user's password.
Minimum required access rights = Read-Write on the user
Returns = void

Table 97. "setPassword" Arguments

Name	Description	Type	Optional
password	new password	string	N

setDisabled

Administratively disable this user account or re-enable an account that was disabled administratively or due to unsuccessful login attempts.
Minimum required access rights = Read-Write on the user
Returns = void

Table 98. "setDisabled" Arguments

Name	Description	Type	Optional
disabled	if true, the account is disabled; if false, the account is enabled	boolean	N

getRedScoreThreshold

Get the user's red score threshold. If the customer has set a red score threshold, the customer's threshold will be returned. Otherwise, the user's red score threshold will be returned.
Minimum required access rights = Read-Only on the user
Returns = user's red score threshold (integer)
No arguments.

setRedScoreThreshold

Set the user's red score threshold. The threshold may not be changed if the user's customer defines a mandatory red score threshold.
Minimum required access rights = Read-Write on the user
Returns = void

Table 99. "setRedScoreThreshold" Arguments

Name	Description	Type	Optional
score	the value to assign to the red score threshold	integer	N

getRoles

Get a list of roles to which this user is assigned.
Minimum required access rights = Read-Only on the user
Returns = a list or table (list of lists) of the user's roles (searchResult)

Table 100. "getRoles" Arguments

Name	Description	Type	Optional
format	Values are "list" (to return a list of object references) and "table" (to return a table of object attributes). Defaults to "list." See "Search Result Formats" on page 9 for a description of how search results are formatted.	string	Y

addRole

Assign a role to this user.
Minimum required access rights = Read-Write on the user
Returns = void

Table 101. "addRole" Arguments

Name	Description	Type	Optional
role	object reference to the role to add	objref	N

removeRole

Remove a role from this user.
Minimum required access rights = Read-Write on the user
Returns = void

Table 102. "removeRole" Arguments

Name	Description	Type	Optional
role	object reference to the role to remove	objref	N

Class Methods

create

Create a new user.
Minimum required access rights = Read-Write on Users

Returns = object reference to the new user instance (objref)

Table 103. "create" Arguments

Name	Description	Type	Optional
firstName	new user's first name	string	N
lastName	new user's last name	string	N
emailAddress	new user's email address	string	N
password	new user's password	string	N
customer	object reference to the customer for which to create user	objref	Y

Class: UserGroup

This class represents user groups in the IP360 system.

Attributes

Table 104. "User Group" Class Attributes

Name	Description	Type	Read Only	Sortable
id	system internal identifier for the user group	integer	Y	N
customer	object reference to the customer to which this user group belongs	objref	Y	N
name	name of the user group	string	N	N

Instance Methods

delete

Delete the user group.
Minimum required access rights = Read-Write on the user group
Returns = void
No arguments.

getMembers

Return a list of the users in the user group.
Minimum required access rights = Read-Only on the user group

Returns = a list or table (list of lists) of the user group's users (searchResult)

Table 105. "getMembers" Arguments

Name	Description	Type	Optional
format	Values are "list" (to return a list of object references) and "table" (to return a table of object attributes). Defaults to "list." See "Search Result Formats" on page 9 for a description of how search results are formatted.	string	Y

addMember

Add a user to the user group.
Minimum required access rights = Read-Write on both the user and the user group
Returns = void

Table 106. "addMember" Arguments

Name	Description	Type	Optional
user	object reference to the user to add to the group	objref	N

removeMember

Remove a user from the user group.
Minimum required access rights = Read-Write on the user group
Returns = void

Table 107. "remove" Arguments

Name	Description	Type	Optional
user	object reference to the user to remove from the group	objref	N

Class Methods

create

Create a new user group.
Minimum required access rights = Read-Write on User Groups

Returns = object reference to the new user group instance (objref)

Table 108. "create" Arguments

Name	Description	Type	Optional
name	name of the new group	string	N
customer	object reference to the customer for which to create user group	objref	Y

Common Class Methods

These class methods can be used for all non-singleton classes.

getAttributes

Returns a list of the names of all attributes defined by the class.
Minimum required access rights = Read-Only on the class of object
Returns = the names of all the attributes defined by the class (list of strings)
No arguments.

getMethods

Returns a list of the names of all methods defined by the class.
Minimum required access rights = Read-Only on the class of object
Returns = the names of all the methods defined by the class (list of strings)
No arguments.

lookup

Looks up a specific instance of the class, given its object ID. If the object exists, the method returns its object reference. Otherwise, it returns the error "ENOENT: Object not found."
Minimum required access rights = Read-Only on the object
Returns = object reference to the instance of the class (objref)

Table 109. "lookup" Arguments

Name	Description	Type	Optional
id	the ID of the object instance being looked up	integer	N

fetch

Returns a list of all instances of the class. `sortBy` *and* `sortOrder` *direct the sorting of results.* `offset` *and* `limit` *limit the scope of results.*
Minimum required access rights = Read-Only on the class of object

Returns = list or table (list of lists) of all the instances of the class (searchResult)

Table 110. "fetch" Arguments

Name	Description	Type	Optional
offset	The number of values that will be skipped before the beginning of the result set. Valid values are zero and greater.	integer	Y
limit	The limit to the number of results returned. Valid values are zero and greater.	integer	Y
sortBy	the sortable attribute which by which to sort the results	string	Y
sortOrder	the order in which to sort the results. Valid value are "ascending" and "descending". If not specified, but `sortBy` is, the sort will be ascending.	string	Y
format	Values are "list" (to return a list of object references) and "table" (to return a table of object attributes). Defaults to "list." See "Search Result Formats" on page 9 for a description of how search results are formatted.	string	Y

search

Returns a list of those instances of the class matching the search query. (See "Search Queries" on page 8.) This method takes the same arguments as and behaves identically to the `fetch` *class method, with the exception that the results are filtered using the search query.*
Minimum required access rights = Read-Only on the class of object
Returns = list or table (list of lists) of all the instances of the class that match the search query (searchResults)

Table 111. "search" Arguments

Name	Description	Type	Optional
query	the query string for the search (See "Search Queries" on page 8 for an explanation of query values.)	string	N
offset	The number of values that will be skipped before the beginning of the result set. Valid values are zero and greater.	integer	Y
limit	The limit to the number of results returned. Valid values are zero and greater.	integer	Y
sortBy	the sortable attribute which by which to sort the results	string	Y

Table 111. "search" Arguments

Name	Description	Type	Optional
sortOrder	the order in which to sort the results. Valid value are "ascending" and "descending". If not specified, but `sortBy` is, the sort will be ascending.	string	Y
format	Valid values are "list" (to return a list of object references) and "table" (to return a table of object attributes). Defaults to "list." See "Search Result Formats" on page 9 for a description of how search results are formatted.	string	Y

Common Instance Methods

These class methods can be used for all instance objects.

getClassName

Returns the name of the class of which the instance is a member.
Minimum required access rights = Read-Only on the class of object
Returns = name of the class of which the instance is a member (string)
No arguments.

getClassObject

Returns an object reference to the class object for the class of which the instance is a member.
Returns = object reference to the class object for the class of which the instance is a member (objref)
No arguments.

getAttributes

*Returns a dictionary that maps the names of the object's attributes to their values.**
Returns = dictionary that maps the names of the object's attributes to their values (XML-RPC "struct")
No arguments.

* For example, the XML-RPC result for a ScanConfiguration object would look like this:

```
<struct>
  <member>
    <name>id</name>
    <value><string>ScanConfiguration.10</string></value>
  </member>
  <member>
    <name>network</name>
    <value><string>Network.1</string></value>
  </member>
  <member>
    <name>scanProfile</name>
    <value><string>ScanProfile.22</string></value>
  </member>
```

```
    <member>
      <name>dp</name>
      <value><string>DP.1</string></value>
    </member>
    <member>
      <name>active</name>
      <value><int>1</int></value>
    </member>
  </struct>
```

getAttribute

Returns the value of the attribute provided as an argument.
Returns = mixed (depends on type of attribute value)

Table 112. "setAttribute" Arguments

Name	Description	Type	Optional
attribute	the attribute whose value is returned	string	N

setAttribute

Changes the value of the attribute provided as an argument.
Returns = void

Table 113. "setAttribute" Arguments

Name	Description	Type	Optional
attribute	the attribute whose value is changed	string	N
value	the new value for the attribute. The data type must match the expected data type and other constraints for the attribute, or the API will return an ETYPE error.	Depends on the attribute	N

SESSION Object Methods

These methods are implemented by the `SESSION` object. For more information on the `SESSION` object, see "The Session Object" on page 6.

getClassByName

Returns the object reference for a class (if it exists), for a class name provided as an argument. If the class does not exist, the method returns the error EBADCLASS: Class not found.
Returns = object reference

Table 114. "getClassByName" Arguments

Name	Description	Type	Optional
name	name of the class	string	N

getEnumValues

Returns a table that maps the enumeration keys to their integer values, for an enumeration name provided as an argument. If the enumeration does not exist, the method returns the error EBADENUM: Enumeration not found.
Returns = table (list of lists) of key-value (integer-enumeration value) pairs*

Table 115. "getEnumValues" Arguments

Name	Description	Type	Optional
name	name of the enumeration	string	N

* For example, calling `getEnumValues` with `name` = "Language" returns:

```
[ [ "English", 1 ],
  [ "Japanese", 2 ]
]
```

The XML-RPC result is:

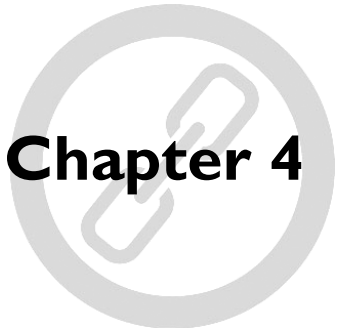
```
<array>
  <value>
    <array>
      <value><string>English</string></value>
      <value><integer>1</integer></value>
    </array>
  </value>
  <value>
    <array>
      <value><string>Japanese</string></value>
      <value><integer>2</integer></value>
    </array>
  </value>
</array>
```

getClassList

Returns a list of the names of all classes supported by the API.
Returns = a list of the names of all the classes support by the API (list of strings)
No arguments.

getUserObject

Returns the object reference for the User object of the user making the API call.
Returns = object reference to the User object of the user making the API call
No arguments.



Chapter 4 IP360 API Python Library

The IP360 API Python library enables integration authors to write simpler, cleaner code for interacting with IP360 by abstracting the details of making XML-RPC requests and decoding the results. This chapter illustrates the use of the Python library when communicating with the IP360 API. Understanding the API's object model (Chapter 4) and the reference information (Chapter 3) is helpful to understanding the examples in this chapter.

Installing the Python Library

To install the IP360 API Python Library:

1. Obtain the library distribution file from nCircle. It will be named `IP360_API-<version>.tar.gz` (or `IP360_API-<version>.zip` for Windows).
2. Extract this archive into a temporary directory.

3. Run `setup.py install` to install the IP360 package into your Python site-packages directory. (Optionally specify `--prefix=<directory>` to select a different installation prefix).
4. If you installed the module in the default location, you should be able to run `pydoc IP360.API` to see the documentation for the library.

Using the Python Library

This section describes the most basic and important tasks performed by a customer's Python program.

Importing the API

Customers' programs using the IP360 API Python library must import the API. Usually, customers can do this simply by including the following line in their program:

```
import IP360.API
```

However, if the API was installed outside Python's default search path, customers must specify the path to the library before the `import` statement above:

```
import sys
sys.path.append('<path to IP360 modules>')
import IP360.API
```

Initiating an API Session

To initiate a new API session, instantiate the `IP360.API.Session` class, specifying the hostname of the IP360 system and the login credentials.

Example.

```
session = IP360.API.Session('ip360.example.com', 'username', 'password')
```

The `session` object is the starting point for all interactions with this IP360 system. It is possible to interact with more than one IP360 system at a time by instantiating more than one API session.

Example.

```
vne1_session = IP360.API.Session('vne1.ip360.example.com', 'username',
    'password')
vne2_session = IP360.API.Session('vne2.ip360.example.com', 'username',
    'password')
```

The SESSION Object

The session object provides the `SESSION` class methods such as `getClassList` and `getUserObject`, as well as access to the classes and enumerated types defined by the API. (See "The IP360 Object Model" on page 5 and "IP360 API Reference" on page 12, respectively, for more information.)

Each API class is defined as a propety of the session object.

Example. To obtain a handle to the User class, reference the User property of the session object:

```
user_class = session.User
```

Users can then look up instances of that class.

Example. To get all users of the system:

```
users = user_class.fetch()
```

Enumerated Types

An API enumerated type (see "Enumerated Data Types" on page 11 for an explanation) can be accessed by referencing the type's name as a property of the `enum` property of the session class.

Example. To obtain the Language enumeration:

```
language_enum = session.enum.Language
```

Each API enumerated type behaves both like a dictionary and like an object with a property named for each enumerated value. A Python program can use whatever notation is most natural or convenient; the values that make up the enumerated type are implemented as special objects that return the value's symbolic name when represented as a string, but they also have an integer value equal to what the API expects.

Example. This is the transcript of an interactive Python session, segmented to show the two different ways the enumeration can be accessed.

```
>>> scan_type = s.enum.ScanType
>>> print s.enum.ScanType
{'Scheduled': 1, 'On Demand': 2, 'Partial': 3}


>>> print scan_type['Scheduled']
IP360.API.ScanType.Scheduled
>>> print int(scan_type['Scheduled'])
1
>>> print scan_type['Scheduled'] == 1
True


>>> print scan_type.Scheduled
IP360.API.ScanType.Scheduled
>>> print int(scan_type.Scheduled)
1
>>> print scan_type.Scheduled == 1
True
```

Instance Objects

Each object accessible using the API is abstracted by an object that is an instance of the corresponding class. For example, if users call `session.getUserObject()`, the method returns an

instance of the IP360.API.User class. If users installed the API library into Python's default location, they can access the documentation for each class using `pydoc`.

Example. To retrieve documentation for the User class:

```
pydoc IP360.API.User.
```

Each object provides access to the attributes and methods defined by its class. For example, a User object has attributes such as `firstName`, `lastName`, `emailAddress`, and `language`. Each attribute can be directly referenced from the object.

Example. To print the first name and last name attributes of a user:

```
my_user = session.getUserObject()
print 'My name is %s %s.' % (my_user.firstName, my_user.lastName)
```

Users can also call methods on the object.

Example. To set a user's password:

```
my_roles = my_user.getRoles()
my_role_names = map(lambda role: role.name, my_roles)
print 'My roles are: %s' % (my_role_names)

my_user.setPassword('my_new_password')
```

Looking Up Instance Objects

Users can look up object instances using the `fetch` and `search` methods defined by class objects. `fetch` returns all of the instances of a class (possibly limited by the `offset` and `limit` parameters), and `search` returns a subset of the instances as selected by the search criteria. The `fetch` and search methods both return a list of instance objects. (See "fetch" on page 68 and "search" on page 69 for more information about these two methods.)

Example. Using `fetch` to retrieve all Network instances:

```
all_networks = session.Network.fetch()
for network in all_networks:
  print 'Network #%d is named "%s"' % (network.id, network.name)
```

Example. Using search to retrieve a specific subset of Network instances:

```
sf_networks = session.Network.search(query="name LIKE '%San
Francisco%')
print 'San Francisco Networks:'
for network in sf_networks:
  print '  %s' % (network.name)
```

Class Methods

Many classes define class methods in addition to `fetch` and `search`.

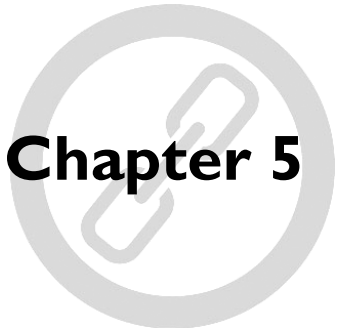
Example. Using the User class's `create` method for creating new users:

```
new_user = session.User.create(firstName="John", lastName="Doe",
                               emailAddress="john.doe@example.com",
                               password="johndoe123")

print 'New user ID is: %d' % (new_user.id)
```

Some classes do not have any instances and implement only class methods. The ASPL class is one such class.

Example. To retrieve the current ASPL version:

```
aspl_version = session.ASPL.getASPLVersion()
print 'Current ASPL database version is %s.' % (aspl_version)
```



Chapter 5 API Error Handling

Error Responses

API requests may result in error responses, either because the parameters of the method were incorrect, or because an IP360 internal fault occurred, preventing the request from being serviced. API errors are returned as XML-RPC fault results.

Example.

```
<?xml version="1.0"?>
<methodResponse>
  <fault>
    <value>
      <struct>
        <member>
          <name>faultCode</name>
          <value><int>807</int></value>
        </member>
        <member>
          <name>faultString</name>
          <value><string>Object not
found</string></value>
        </member>
```

```
        </struct>
      </value>
    </fault>
</methodResponse>
```

If using the Python libxmlrpc XML-RPC implementation, an `xmlrpclib.Fault` exception will be raised when the API reports a fault.

API Fault Codes

XML-RPC encodes fault types as integers. This is the list of API fault codes indexed by their XML-RPC fault number. Fault code 899 indicates that an error occurred while processing the method call, either due to invalid parameters or due to an internal problem within IP360. All code 899 error messages will be of the form:

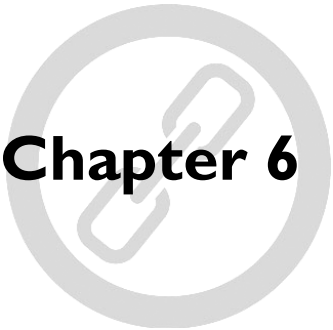
```
<error code>: <error message>
```

Example.

```
ENOENT: Object does not exist
EBADPASSWORD: Your password must contain at least 1 number
EPERM: Read permission denied to Network #6
```

Table 116. API Fault Codes

Fault #	Fault Description
800	Invalid login (wrong username and/or password)
801	Unsupported API version
802	Invalid session ID
810	Missing required XML-RPC parameter
811	XML-RPC parameter of wrong type
812	Internal error
899	API method call error



Chapter 6 Sample Code for Basic API Tasks

This chapter provides sample code for basic IP360 tasks through the IP360 API. The code is written in Python, but it does not take advantage of the IP360 API Python library, as described in "IP360 API Python Library" on page 73.

Create a New IP360 User

```
# This sample Python script creates a new IP360 user
# and resets the initial password.

# Import the xmlrpclib module
import xmlrpclib

# Invoke a method on the XML-RPC server of IP360
def callMethod(cookie, object, method, params):
  try:
      result = c.call(cookie, object, method,
                      params, {'overall': 30000, 'priority': 0})
      return result
  except xmlrpclib.Fault, fault:
      raise xmlrpclib.Fault, fault

# Create a proxy server instance
c = xmlrpclib.ServerProxy('https://<IP360 VnE address>/api/index.ice')

# Create a session with the IP360 API supplying user credentials
try:
    Cookie = c.login(1, 0, 'jdoe@example.com', 'password')
except RuntimeError:
    print ' failure'

# Invoke a method to create a new IP360 user
try:
    userInstance = callMethod(Cookie, 'class.User', 'create',
                              {'lastName':'Doe','firstName':'John',
                               'emailAddress':'jdoe@example.com',
                               'password':'changeme'})
   # Method call to reset the password
    Status = callMethod(Cookie, userInstance,
                        'setPassword', {'password':'ip360'})
    print Status
except xmlrpclib.Fault, fault:
    print fault.faultCode
    print fault.faultString

# End the session with the IP360 API
try:
    c.logout(Cookie)
except xmlrpclib.Fault, fault:
   print fault.faultCode
except RuntimeError:
    print 'Logout failed'
```

Create and Define a Network

```
# This sample Python script creates a new network
# and defines that network.

# Import the xmlrpclib module
import xmlrpclib

# Invoke a method on the XML-RPC Server of IP360
def callMethod(cookie, object, method, params):
  try:
      print method
      result = c.call(cookie, object, method, params,
                      {'overall': 30000, 'priority': 0})
      print result
      return result
  except xmlrpclib.Fault, fault:
      raise xmlrpclib.Fault, fault

# Create a proxy server instance
c = xmlrpclib.ServerProxy('https://<IP360 VnE address>/api/index.ice')

# Create a session with IP360 APIsupplying user credentials
try:
    Cookie = c.login(1, 0, 'jdoe@example.com', 'password')
    print Cookie
except RuntimeError:
    print ' failure'

# Invoke a method to create a new network and define the network
try:
    netObject = callMethod(Cookie, 'class.Network', 'create',
                           {'name':'Test Network'})
    # Include a Network Definition
    ips = [ ]
    ips.append('10.1.1.0/24')
    result = callMethod(Cookie, netObject, 'addIncludes',
{'addrs':ips})
except xmlrpclib.Fault, fault:
    print fault.faultCode
    print fault.faultString

# End the session with the IP360 API
try:
    c.logout(Cookie)
except xmlrpclib.Fault, fault:
    print fault.faultCode
except RuntimeError:
    print 'Logout failed'
```

Scan on Demand (Manually Start a Scan)

```
# This sample Python script initiates a Scan on Demand.

# Import the xmlrpclib module #
import xmlrpclib

# Function that invokes a method on IP360's XML-RPC server
def callMethod(cookie, object, method, params):
  try:
      print method
      result = c.call(cookie, object, method, params, {'overall': 30000,
                                                       'priority': 0 })
      print result
      return result
  except xmlrpclib.Fault, fault:
         raise xmlrpclib.Fault, fault

################################
# Program execution begins here #
################################

# Create a proxy server instance
c = xmlrpclib.ServerProxy('https://<ip360 host>/api/index.ice')

# Create a session with IP360 API, supplying user credentials
try:
    Cookie = c.login(1, 0, 'johndoe@example.com', 'password')
    print Cookie
except RuntimeError:
    print ' failure'

try:
    # Search for the Device Profiler to use in the scan
    dplist = callMethod(Cookie, 'class.DP', 'search', {'query':'name =
                                                       \'dp.example.com\''})
    for object in dplist:
        dpObjectRef = object

    # Search for the Scan Profile to use in the scan
    splist = callMethod(Cookie, 'class.ScanProfile', 'search',
                     {'query':'name = \'nCircle: Full Scan\''})
    for object in splist:
        spObjectRef = object

    # Search for the Network to scan
    list = callMethod(Cookie, 'class.Network', 'search', {'query':'name =
                                                          \'ABC Network\''})
    for object in list:
        networkObjectRef = object

    # Invoke IP360 API method call to start the Scan On Demand
    netObject = callMethod(Cookie, dpObjectRef, 'startScan',
                           {'network':networkObject,
                            'scanProfile':spObjectRef})
except xmlrpclib.Fault, fault:
   print fault.faultCode
    print fault.faultString
```

```
# End session with IP360 API
try:
    c.logout(Cookie)
except xmlrpclib.Fault, fault:
    print fault.faultCode
except RuntimeError:
    print 'Logout failed'
```

Appendix XML-RPC Protocol Details

This chapter describes the IP360 API's HTTPS XML-RPC interface. It provides examples of API requests in XML. However, users can use the IP360 API Python library (as described in "IP360 API Python Library" on page 73) or any other language they prefer to communicate with the API.

XML-RPC is a standard, like CORBA and DCOM, that enables disparate systems distributed over a network to communicate with each other. RPC enables a client to invoke a remote function call just as it would a local function call; the data transported between client and server is encoded in XML. Because XML-RPC is a standard, implementations are available in C, C++, Java, PHP, Python. More information about the XML-RPC standard is available at http://www.xmlrpc.org.

Note: The maximum request made to IP360 API can be 64 KB long.

Writing an XML-RPC API Client

To write an API client, users need only a programming language of their choice and the XML-RPC implementation library for that language.

Example. Here is an XML-RPC API client in Python:

```
# This example illustrates a simple XML-RPC client implementation.
#  This client connects to example.com and accesses its server
#  implementation by invoking a method to print the state name.

# Import the xmlrpclib client module
import xmlrpclib

# Establish a connection to server and create a server proxy instance
server = xmlrpclib.ServerProxy("http://example.com")

# Invoke a method on the remote server and print the result.
try:
    print server.examples.getStateName(5)
except Error, v:
    print "ERROR", v
```

Example. Here is an XML-RPC API client in ActionScript (Adobe Flex):

```
//Using as3-rpclib to get an XMLRPCObject

var service:XMLRPCObject = new XMLRPCObject();
service.destination = "/api/index.ice";
service.endpoint = "https://a.b.c.d"; //VnE IP address

//Call to login, with result and fault function pointers

var token:AsyncToken = service.call("login", 1, 0, userName,
userPass);
var responder:IResponder = new Responder(result, fault)
token.addResponder(responder);

//Example -- get scan profiles; struct is an impl of
//IXMLRPCStruct (see below)

var token:AsyncToken =
    service.call("call",
                 sessionID,
                 "class.ScanProfile",
                 "search",
                 new QueryValueXmlRpcStruct(searchCriteria));
var responder:IResponder = new Responder(result, fault);
token.addResponder(responder);


//Example of how to impl struct object

public class QueryValueXmlRpcStruct implements IXMLRPCStruct {
    public var value:String;

    public function QueryValueXmlRpcStruct(value:String) {
        this.value = value;
```

```
    }

    public function getPropertyData():* {
        return {query:value};
    }
}
```

IP360 API XML-RPC Requests

All interaction with the IP360 API takes place using the XML-RPC protocol over an HTTPS connection to the IP360 VnE. (The protocol specification and links to implementations of the protocol for many common programming languages can be found at http://www.xmlrpc.com/.)

Accessing the API

The XML-RPC URL for the IP360 API is: https://<customer VnE Manager hostname>/api/

Logging into the API

To log in to the API, call the `login` XML-RPC method, specifying these parameters: (integer) <API Major Version>, (integer) <API Minor Version>, (string) <Username>, (string) <Password>

The current API major version is 1, and the current API minor version is 0. IP360 API usernames and passwords are the same as those used for the IP360 web user interface. IP360 users have the same permissions when using the API as when using the UI.

Example. To perform an API login request for a user with username "user@example.com" and password "mypassword", send this request:

```
<?xml version="1.0"?>
<methodCall>
  <methodName>login</methodName>
  <params>
    <param>
      <value>
        <int>1</int>
      </value>
    </param>
    <param>
      <value>
        <int>0</int>
      </value>
    </param>
    <param>
      <value>
        <string>user@example.com</string>
      </value>
    </param>
    <param>
```

```
        <value>
          <string>mypassword</string>
        </value>
      </param>
    </params>
  </methodCall>
```

The API will return an XML-RPC response that looks like:

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <string>SESSION_ID</string>
    </param>
  </params>
</methodResponse>
```

The `SESSION_ID` will be a random string of characters, for example, `17f8dddfb9c74f5cada70492474ce798`. This ID must be passed to each subsequent API request. Like an IP360 UI login session does, an API login session will time out after the configured period of inactivity.

Logging out of the API

When users are done with an API session, they should close it using the `logout` XML-RPC method. The `logout` XML-RPC method takes a single argument: (string) the session ID of the session to cancel.

Example. To log out from session `17f8dddfb9c74f5cada70492474ce798`, send this request:

```
<?xml version="1.0"?>
<methodCall>
  <methodName>logout</methodName>
  <params>
    <param>
      <value>
        <string>17f8dddfb9c74f5cada70492474ce798</string>
      </value>
    </param>
  </params>
</methodCall>
```

The API will return the XML-RPC response:

```
<?xml version="1.0"?>
<methodResponse>
  <params/>
</methodResponse>
```

Making IP360 API Calls

After a session is established with the `login` method, users can call API methods. API method calls take the form of an XML-RPC method call to the method named "call" that take four arguments: (string) `Session ID`, (string) `Object Reference`, (string) `Method Name`, (struct) `Method`.

Using Method Arguments

To illustrate the use of arguments in a method, following are sample requests that perform a method call with certain parameters. These are the XML-RPC requests (note that `SESSION_ID` must be replaced by the Session ID returned from the `login` XML-RPC method):

1. This sample performs a search for all Appliance objects with matching Type (in this case). This is a typical search-type call: the value of <name> in the struct is always "query" and the value of <value> is the query to perform (for example , Type =1):

Object	`APPLIANCE`
Method	`search`
Arguments	`Type=1`

```
<?xml version="1.0"?>
<methodCall>
  <methodName>call</methodName>
  <params>
    <param>
      <value><string>SESSION_ID</string></value>
    </param>
    <param>
      <value><string>class.Appliance</string></value>
    </param>
    <param>
      <value><string>search</string></value>
    </param>
    <param>
      <value>
        <struct>
          <member>
            <name>query</name>
            <value><string>type=1</string></value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

2. This sample is a typical `getAttributes` query, which will return all the attributes of a given object. Arguments for `getAttributes` are:
   - the ID of the Object whose attributes you want to return (User.1) IDs are comprised of the Class name dot-separated with the object's unique ID.
   - search – always the second argument in a `getAttributes` call

- getAttributes – always the third argument in a `getAttributes` call

Object	`USER`
Method	`getAttributes`
Arguments	`Classname.id` `search`

```
<?xml version="1.0"?>
<methodCall>
  <methodName>call</methodName>
  <params>
    <param>
      <value><string>SESSION_ID</string></value>
    </param>
    <param>
      <value><string>User.1</string></value>
    </param>
    <param>
      <value><string>search</string></value>
    </param>
    <param>
       <value><string>getAttributes</string></value>
    </param>
  </params>
</methodCall>
```

3. This sample is a typical search query, which will return either all matching Object IDs (Classname.id) or complete objects. The format for the results is determined by the search result type, either "list" or "table" (see “Search Result Formats” on page 9). This example returns all ScanProfiles whose "active" field is true.:

Object	`ScanProfile`
Method	`search`
Arguments	`query`

```
<?xml version="1.0"?>
<methodCall>
  <methodName>call</methodName>
  <params>
    <param>
      <value><string>SESSION_ID</string></value>
    </param>
    <param>
      <value><string>class.ScanProfile</string></value>
    </param>
    <param>
      <value><string>search</string></value>
    </param>
    <param>
      <value>
        <struct>
          <member>
            <name>query</name>
            <value><string>active</string></value>
```

```
            </member>
          </struct>
        </value>
      </param>
    </params>
  </methodCall>
```

4. This sample returns the ID of the User Object, containing user information such as first/last name, email, etc. It does not take any arguments:

Object	SESSION
Method	getUserObject
Arguments	none

```
<?xml version="1.0"?>
<methodCall>
  <methodName>call</methodName>
  <params>
    <param>
      <value><string>SESSION_ID</string></value>
    </param>
    <param>
      <value><string>SESSION</string></value>
    </param>
    <param>
      <value><string>getUserObject</string></value>
    </param>
    </methodCall>
```