
WSDiscovery

Release 2.0

L.A. Fernando, Andrei Kopats, Pieter Jordaan, Petri Savolainen, M

Sep 22, 2023

TABLE OF CONTENTS:

1	WS-Discovery in a nutshell	3
1.1	WS-Discovery message semantics	3
1.2	WS-Discovery terms	3
2	Command-line utilities	5
2.1	wsdiscover	5
2.2	wspublish	5
3	Framework for discovery applications	7
3.1	Service publishing implementation	7
3.2	Service discovery implementation	7
3.3	Networking daemon base classes	8
3.4	WS-Discovery message facilities	10
3.5	Service representation	13
3.6	Service Scope representation	13
3.7	QName representation	14
3.8	URI parser for scope matching	14
3.9	UDP communications handling	14
3.10	Namespace definitions	15
4	Indices and tables	17
	Python Module Index	19
	Index	21

This package provides a short introduction to WS-Discovery, simple command-line tools for discovering and publishing services, and a framework for building WS-Discovery service publishing or discovery applications.

WS-DISCOVERY IN A NUTSHELL

WS-Discovery (Web Services Discovery) is a standard widely used by network cameras and other connected devices such as printers. It is based on uni- and multicast [SOAP over UDP](#).

1.1 WS-Discovery message semantics

WS-Discovery defines the following **message semantics** for managing and discovering the availability of networked services.

See [WS-Discovery terms](#) for explanation of some common terms.

Hello A service *must* send a one-way multicast Hello message when it joins a network, or its metadata changes.

Probe To discover services, optionally limited to a particular service type or scope, a client sends a Probe message. Probe can be unicast or multicast.

Probe match When a service receives a matching Probe, it *must* respond with a Probe Match message.

Resolve A client may send a one-way multicast Resolve message to locate service address(es).

Resolve match When a service matches a Resolve message, it *must* respond with a unicast Resolve Match message.

Bye A service *should* send a one-way multicast Bye message when preparing to leave a network.

1.2 WS-Discovery terms

Action In WS-Addressing, Action (URI) identifies the semantics of the message.

QName A name used in XML is uniquely qualified when it's associated with a namespace (URI) it belongs to. This package provides a [QName representation](#) implementation.

Scope Scopes are common identifiers used for organizing web services into logical groups, serving a purpose similar to that of categories or tags. This package provides a [Service Scope representation](#) implementation.

EPR An Endpoint reference is an URI (Uniform Resource Identifier) that identifies a SOAP resource such as a WS-Discovery service. A EPR is included in a [Probe match](#) message. A [Resolve](#) message can then be used to retrieve actual network address for service.

Envelope SOAP messages are wrapped in so-called envelopes. This package provides a [WS-Discovery message envelope & factories](#) implementation.

For details, see the full [WS-Discovery specification](#).

COMMAND-LINE UTILITIES

There are two command-line tools available for discovering and publishing services.

2.1 wsdiscover

Discover services using WS-Discovery

```
wsdiscover [OPTIONS]
```

Options

- s, --scope** <scope>
Full scope URI, eg. onvif://www.onvif.org/Model/
- a, --address** <address>
Service address
- p, --port** <port>
Service port
- l, --loglevel** <loglevel>
Log level; one of INFO, DEBUG, WARNING, ERROR
- c, --capture** <capture>
Capture messages to a file

2.2 wspublish

Publish services using WS-Discovery

```
wspublish [OPTIONS]
```

Options

- s, --scope** <scope>
Full scope URI, eg. `onvif://www.onvif.org/Model/`
- t, --typename** <typename>
Qualified type name, eg. `https://myservicesns:myservice_type`
- a, --address** <address>
Service IP address
- p, --port** <port>
Service port
- l, --loglevel** <loglevel>
Log level; one of INFO, DEBUG, WARNING, ERROR
- c, --capture** <capture>
Capture messages to a file

FRAMEWORK FOR DISCOVERY APPLICATIONS

Reusable facilities for implementing networked service publishing and discovery applications.

3.1 Service publishing implementation

```
class wsdiscovery.publishing.ThreadedWSPublishing (**kwargs)
    Bases:      wsdiscovery.threaded.ThreadedNetworking,  wsdiscovery.publishing.
                Publishing, wsdiscovery.daemon.Daemon

    threaded service publishing

    clearLocalServices ()
        send Bye messages for the services and remove them

    publishService (types, scopes, xAddrs)
        Publish a service with the given TYPES, SCOPES and XAddrs (service addresses)

        if xAddrs contains item, which includes {ip} pattern, one item per IP address will be sent

    start ()
        start networking - should be called before using other methods

    stop ()
        cleans up and stops networking
```

3.2 Service discovery implementation

```
class wsdiscovery.discovery.ThreadedWSDiscovery (**kwargs)
    Bases:      wsdiscovery.daemon.Daemon,      wsdiscovery.discovery.Discovery,
                wsdiscovery.threaded.ThreadedNetworking

    Full threaded service discovery implementation

    clearRemoteServices ()
        clears remotely discovered services

    searchServices (types=None, scopes=None, address=None, port=None, timeout=3)
        search for services given the TYPES and SCOPES in a given TIMEOUT

    setRemoteServiceByeCallback (cb)
        Set callback, which will be called when new service appeared online and sent Hi message Service is passed
        as a parameter to the callback Set None to disable callback
```

setRemoteServiceHelloCallback (*cb*, *types=None*, *scopes=None*)

Set callback, which will be called when new service appeared online and sent Hi message

typesFilter and *scopesFilter* might be list of types and scopes. If filter is set, callback is called only for Hello messages, which match filter

Set None to disable callback

start ()

start networking - should be called before using other methods

stop ()

cleans up and stops networking

3.3 Networking daemon base classes

The framework decouples WS-Discovery messaging functionality from the actual networking system implementation.

A set of base classes are provided for creating networked service discovery and/or publishing implementations:

3.3.1 Generic networking base daemon

Generic networking-agnostic WS-Discovery messaging daemon mixin implementation.

class `wsdiscovery.daemon.Daemon` (*uuid_=None*, *capture=None*, *ttl=1*, ***kwargs*)

generic WS-Discovery messaging daemon implementation

_sendBye (*service*)

_sendHello (*service*)

_sendProbe (*types=None*, *scopes=None*, *address=None*, *port=None*)

_sendProbeMatch (*services*, *relatesTo*, *addr*)

_sendResolve (*epr*)

_sendResolveMatch (*service*, *relatesTo*, *addr*)

envReceived (*env*, *addr*)

3.3.2 Threaded networking base classes

Threaded networking facilities for implementing threaded WS-Discovery daemons.

class `wsdiscovery.threaded.AddressMonitorThread` (*wsd*)

trigger address change callbacks when local service addresses change

_updateAddrs ()

run ()

Method representing the thread's activity.

You may override this method in a subclass. The standard `run()` method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the `args` and `kwargs` arguments, respectively.

class `wsdiscovery.threaded.NetworkingThread` (*observer*, *capture=None*)

```

static _createMulticastInSocket ()
static _createMulticastOutSocket (addr, ttl)
static _makeMreq (addr)
_recvMessages ()
_sendMsg (msg)
_sendPendingMessages ()
    Method sleeps, if nothing to do
addMulticastMessage (env, addr, port, initialDelay=0)
addSourceAddr (addr)
    None means 'system default'
addUnicastMessage (env, addr, port, initialDelay=0)

```

```

join ()

```

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out.

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

```

removeSourceAddr (addr)

```

```

run ()

```

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

```

start ()

```

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

```

class wsdiscovery.threaded.ThreadedNetworking (**kwargs)
    handle threaded networking start & stop, address add/remove & message sending
    _startThreads ()
    _stopThreads ()
    addSourceAddr (addr)
    removeSourceAddr (addr)

```

```
sendMulticastMessage (env, initialDelay=0)  
    handle multicast message sending  
  
sendUnicastMessage (env, host, port, initialDelay=0)  
    handle unicast message sending  
  
start ()  
    start networking - should be called before using other methods  
  
stop ()  
    cleans up and stops networking  
  
class wsdiscovery.threaded._StoppableDaemonThread  
    Stoppable daemon thread.  
  
    run() method shall exit, when self._quitEvent.wait() returned True  
  
schedule_stop ()  
    Schedule stopping the thread. Use join() to wait, until thread really has been stopped
```

3.4 WS-Discovery message facilities

3.4.1 WS-Discovery message envelope & factories

SOAP envelope implementation.

```
class wsdiscovery.envelope.SoapEnvelope  
    envelope implementation  
  
    getAction ()  
  
    getEPR ()  
  
    getInstanceId ()  
  
    getMessageId ()  
  
    getMessageNumber ()  
  
    getMetadataVersion ()  
  
    getProbeResolveMatches ()  
  
    getRelatesTo ()  
  
    getRelationshipType ()  
  
    getReplyTo ()  
  
    getScopes ()  
  
    getSequenceId ()  
  
    getTo ()  
  
    getTypes ()  
  
    getXAddrs ()  
  
    setAction (action)  
  
    setEPR (epr)  
  
    setInstanceId (instanceId)
```

```

setMessageId (messageId)
setMessageNumber (messageNumber)
setMetadataVersion (metadataVersion)
setProbeResolveMatches (probeResolveMatches)
setRelatesTo (relatesTo)
setRelationshipType (relationshipType)
setReplyTo (replyTo)
setScopes (scopes)
setSequenceId (sequenceId)
setTo (to)
setTypes (types)
setXAddr (xAddr)

```

3.4.2 WS-Discovery messages (de)serialization

Functions to serialize and deserialize messages between SOAP envelope & string representations

```

wsdiscovery.message.createSOAPMessage (env)
    serialize SOAP envelopes into XML strings

wsdiscovery.message.parseSOAPMessage (data, ipAddr)
    deserialize XML message strings into SOAP envelope objects

```

Serialization & deserialization functions for each message:

Bye message (de)serialization

Serialize & parse WS-Discovery Bye SOAP messages

```

wsdiscovery.actions.bye.constructBye (service)
    construct an envelope that represents a Bye message

wsdiscovery.actions.bye.createByeMessage (env)
    serialize a SOAP envelope object into a string

wsdiscovery.actions.bye.parseByeMessage (dom)
    parse a XML message into a SOAP envelope object

```

Hello message (de)serialization

Serialize & parse WS-Discovery Hello SOAP messages

```

wsdiscovery.actions.hello.constructHello (service)
    construct an envelope that represents a Hello message

wsdiscovery.actions.hello.createHelloMessage (env)
    serialize a SOAP envelope object into a string

wsdiscovery.actions.hello.parseHelloMessage (dom)
    parse a XML message into a SOAP envelope object

```

Probe message (de)serialization

Serialize & parse WS-Discovery Probe SOAP messages

`wsdiscovery.actions.probe.constructProbe` (*types, scopes*)
construct an envelope that represents a `Probe` message

`wsdiscovery.actions.probe.createProbeMessage` (*env*)
serialize a SOAP envelope object into a string

`wsdiscovery.actions.probe.parseProbeMessage` (*dom*)
parse a XML message into a SOAP envelope object

Probe match message (de)serialization

Serialize & parse WS-Discovery Probe Match SOAP messages

`wsdiscovery.actions.probematch.constructProbeMatch` (*services, relatesTo*)
construct an envelope that represents a `Probe Match` message

`wsdiscovery.actions.probematch.createProbeMatchMessage` (*env*)
serialize a SOAP envelope object into a string

`wsdiscovery.actions.probematch.parseProbeMatchMessage` (*dom*)
parse a XML message into a SOAP envelope object

Resolve message (de)serialization

Serialize & parse WS-Discovery Resolve SOAP messages

`wsdiscovery.actions.resolve.constructResolve` (*epr*)
construct an envelope that represents a `Resolve` message

`wsdiscovery.actions.resolve.createResolveMessage` (*env*)
serialize a SOAP envelope object into a string

`wsdiscovery.actions.resolve.parseResolveMessage` (*dom*)
parse a XML message into a SOAP envelope object

Resolve match message (de)serialization

Serialize & parse WS-Discovery Resolve Match SOAP messages

`wsdiscovery.actions.resolvematch.constructResolveMatch` (*service, relatesTo*)
construct an envelope that represents a `Resolve Match` message

`wsdiscovery.actions.resolvematch.createResolveMatchMessage` (*env*)
serialize a SOAP envelope object into a string

`wsdiscovery.actions.resolvematch.parseResolveMatchMessage` (*dom*)
parse a XML message into a SOAP envelope object

See also:

WS-Discovery message envelope & factories

3.5 Service representation

Discoverable WS-Discovery service.

class `wsdiscovery.service.Service` (*types, scopes, xAddr, epr, instanceId*)

A web service representation implementation

```

getEPR ()
    get endpoint reference

getInstanceId ()

getMessageNumber ()

getMetadataVersion ()

getScopes ()
    get the service scopes

getTypes ()
    get service types

getXAddr ()
    get service network address

incrementMessageNumber ()

setEPR (epr)
    set endpoint reference

setInstanceId (instanceId)

setMessageNumber (messageNumber)

setMetadataVersion (metadataVersion)

setScopes (scopes)
    set the service scopes

setTypes (types)
    set service types

setXAddr (xAddr)
    set service network address

```

3.6 Service Scope representation

Service scopes are used to constrain service discovery.

class `wsdiscovery.scope.Scope` (*value, matchBy=None*)

Service scope implementation.

```

getMatchBy ()

getQuotedValue ()

getValue ()

```

3.7 QName representation

Qualified name support; see e.g. <https://en.wikipedia.org/wiki/QName>

```
class wsdiscovery.qname.QName (namespace, localname, namespace_prefix=None)
    Qualified name implementation

    getFullname ()
    getLocalname ()
    getNamespace ()
    getNamespacePrefix ()
```

3.8 URI parser for scope matching

Module with URI implementation that supports service scope matching

```
class wsdiscovery.uri.URI (uri)
    URI implementation with additional functionality useful for service scope matching

    getAuthority ()
    getPath ()
    getPathExQueryFragment ()
    getScheme ()
```

3.9 UDP communications handling

UDP (User Datagram Protocol) message implementation that helps with management of unicast/multicast semantics and repeat & delay handling.

WS-Discovery spec dictates that the example algorithm provided in SOAP-over-UDP spec is to be used; see

http://docs.oasis-open.org/ws-dd/soapoverudp/1.1/os/wsdd-soapoverudp-1.1-spec-os.html#_Toc229451838

```
class wsdiscovery.udp.UDPMessage (env, addr, port, msgType, initialDelay=0)
    UDP message management implementation

    MULTICAST = 'multicast'
    UNICAST = 'unicast'

    canSend ()
    getAddr ()
    getEnv ()
    getPort ()
    isFinished ()
    msgType ()
    refresh ()
```

3.10 Namespace definitions

SOAP & WS-Discovery XML namespaces used by the package

`wsdiscovery.namespaces.NS_ACTION_BYE` = `'http://schemas.xmlsoap.org/ws/2005/04/discovery/Bye'`
Bye message namespace

`wsdiscovery.namespaces.NS_ACTION_HELLO` = `'http://schemas.xmlsoap.org/ws/2005/04/discovery/Hello'`
Hello message namespace

`wsdiscovery.namespaces.NS_ACTION_PROBE` = `'http://schemas.xmlsoap.org/ws/2005/04/discovery/Probe'`
Probe message namespace

`wsdiscovery.namespaces.NS_ACTION_PROBE_MATCH` = `'http://schemas.xmlsoap.org/ws/2005/04/discovery/ProbeMatch'`
Probe match message namespace

`wsdiscovery.namespaces.NS_ACTION_RESOLVE` = `'http://schemas.xmlsoap.org/ws/2005/04/discovery/Resolve'`
Resolve message namespace

`wsdiscovery.namespaces.NS_ACTION_RESOLVE_MATCH` = `'http://schemas.xmlsoap.org/ws/2005/04/discovery/ResolveMatch'`
Resolve match message namespace

`wsdiscovery.namespaces.NS_ADDRESSING` = `'http://schemas.xmlsoap.org/ws/2004/08/addressing'`
Addressing namespace

`wsdiscovery.namespaces.NS_DISCOVERY` = `'http://schemas.xmlsoap.org/ws/2005/04/discovery'`
Discovery namespace

`wsdiscovery.namespaces.NS_SOAPENV` = `'http://www.w3.org/2003/05/soap-envelope'`
SOAP envelope namespace

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

W

- `wsdiscovery.actions.bye`, [11](#)
- `wsdiscovery.actions.hello`, [11](#)
- `wsdiscovery.actions.probe`, [12](#)
- `wsdiscovery.actions.probematch`, [12](#)
- `wsdiscovery.actions.resolve`, [12](#)
- `wsdiscovery.actions.resolvematch`, [12](#)
- `wsdiscovery.daemon`, [8](#)
- `wsdiscovery.envelope`, [10](#)
- `wsdiscovery.message`, [11](#)
- `wsdiscovery.namespaces`, [15](#)
- `wsdiscovery.qname`, [14](#)
- `wsdiscovery.scope`, [13](#)
- `wsdiscovery.service`, [13](#)
- `wsdiscovery.threaded`, [8](#)
- `wsdiscovery.udp`, [14](#)
- `wsdiscovery.uri`, [14](#)

Symbols

<code>_StoppableDaemonThread</code> (class in <code>wsdiscovery.threaded</code>), 10	<code>wspublish</code> command line option, 6
<code>_createMulticastInSocket()</code> (<code>wsdiscovery.threaded.NetworkingThread</code> static method), 8	<code>--capture <capture></code> wsdiscover command line option, 5 wspublish command line option, 6
<code>_createMulticastOutSocket()</code> (<code>wsdiscovery.threaded.NetworkingThread</code> static method), 9	<code>--loglevel <loglevel></code> wsdiscover command line option, 5 wspublish command line option, 6
<code>_makeMreq()</code> (<code>wsdiscovery.threaded.NetworkingThread</code> static method), 9	<code>--port <port></code> wsdiscover command line option, 5 wspublish command line option, 6
<code>_recvMessages()</code> (<code>wsdiscovery.threaded.NetworkingThread</code> method), 9	<code>--scope <scope></code> wsdiscover command line option, 5 wspublish command line option, 6
<code>_sendBye()</code> (<code>wsdiscovery.daemon.Daemon</code> method), 8	<code>--typename <typename></code> wspublish command line option, 6
<code>_sendHello()</code> (<code>wsdiscovery.daemon.Daemon</code> method), 8	<code>-a</code> wsdiscover command line option, 5 wspublish command line option, 6
<code>_sendMsg()</code> (<code>wsdiscovery.threaded.NetworkingThread</code> method), 9	<code>-c</code> wsdiscover command line option, 5 wspublish command line option, 6
<code>_sendPendingMessages()</code> (<code>wsdiscovery.threaded.NetworkingThread</code> method), 9	<code>-l</code> wsdiscover command line option, 5 wspublish command line option, 6
<code>_sendProbe()</code> (<code>wsdiscovery.daemon.Daemon</code> method), 8	<code>-p</code> wsdiscover command line option, 5 wspublish command line option, 6
<code>_sendProbeMatch()</code> (<code>wsdiscovery.daemon.Daemon</code> method), 8	<code>-s</code> wsdiscover command line option, 5 wspublish command line option, 6
<code>_sendResolve()</code> (<code>wsdiscovery.daemon.Daemon</code> method), 8	<code>-t</code> wspublish command line option, 6
<code>_sendResolveMatch()</code> (<code>wsdiscovery.daemon.Daemon</code> method), 8	A
<code>_startThreads()</code> (<code>wsdiscovery.threaded.ThreadedNetworking</code> method), 9	Action, 3
<code>_stopThreads()</code> (<code>wsdiscovery.threaded.ThreadedNetworking</code> method), 9	<code>addMulticastMessage()</code> (<code>wsdiscovery.threaded.NetworkingThread</code> method), 9
<code>_updateAddrs()</code> (<code>wsdiscovery.threaded.AddressMonitorThread</code> method), 8	<code>AddressMonitorThread</code> (class in <code>wsdiscovery.threaded</code>), 8
<code>--address <address></code> wsdiscover command line option, 5	<code>addSourceAddr()</code> (<code>wsdiscovery.threaded.NetworkingThread</code> method), 9

9
addSourceAddr() (wsdiscovery.threaded.ThreadedNetworking
9
addUnicastMessage() (wsdiscovery.threaded.NetworkingThread
9

B

Bye, 3

C

canSend() (wsdiscovery.udp.UDPMessage method), 14
clearLocalServices() (wsdiscovery.publishing.ThreadedWSPublishing
method), 7
clearRemoteServices() (wsdiscovery.discovery.ThreadedWSDiscovery
7
constructBye() (in module wsdiscovery.actions.bye), 11
constructHello() (in module wsdiscovery.actions.hello), 11
constructProbe() (in module wsdiscovery.actions.probe), 12
constructProbeMatch() (in module wsdiscovery.actions.probematch), 12
constructResolve() (in module wsdiscovery.actions.resolve), 12
constructResolveMatch() (in module wsdiscovery.actions.resolvematch), 12
createByeMessage() (in module wsdiscovery.actions.bye), 11
createHelloMessage() (in module wsdiscovery.actions.hello), 11
createProbeMatchMessage() (in module wsdiscovery.actions.probematch), 12
createProbeMessage() (in module wsdiscovery.actions.probe), 12
createResolveMatchMessage() (in module wsdiscovery.actions.resolvematch), 12
createResolveMessage() (in module wsdiscovery.actions.resolve), 12
createSOAPMessage() (in module wsdiscovery.message), 11

D

Daemon (class in wsdiscovery.daemon), 8

E

Envelope, 3
envReceived() (wsdiscovery.daemon.Daemon
method), 8

EPR, 3

G

getAction() (wsdiscovery.envelope.SoapEnvelope
method), 10
getAddr() (wsdiscovery.udp.UDPMessage method), 14
getAuthority() (wsdiscovery.uri.URI method), 14
getEnv() (wsdiscovery.udp.UDPMessage method), 14
getEPR() (wsdiscovery.envelope.SoapEnvelope
method), 10
getEPR() (wsdiscovery.service.Service method), 13
getFullname() (wsdiscovery.qname.QName
method), 14
getInstanceId() (wsdiscovery.envelope.SoapEnvelope method), 10
getInstanceId() (wsdiscovery.service.Service
method), 13
getLocalname() (wsdiscovery.qname.QName
method), 14
getMatchBy() (wsdiscovery.scope.Scope method), 13
getMessageId() (wsdiscovery.envelope.SoapEnvelope method), 10
getMessageNumber() (wsdiscovery.envelope.SoapEnvelope method), 10
getMessageNumber() (wsdiscovery.service.Service
method), 13
getMetadataVersion() (wsdiscovery.envelope.SoapEnvelope method), 10
getMetadataVersion() (wsdiscovery.service.Service method), 13
getNamespace() (wsdiscovery.qname.QName
method), 14
getNamespacePrefix() (wsdiscovery.qname.QName method), 14
getPath() (wsdiscovery.uri.URI method), 14
getPathExQueryFragment() (wsdiscovery.uri.URI method), 14
getPort() (wsdiscovery.udp.UDPMessage method), 14
getProbeResolveMatches() (wsdiscovery.envelope.SoapEnvelope method), 10
getQuotedValue() (wsdiscovery.scope.Scope
method), 13
getRelatesTo() (wsdiscovery.envelope.SoapEnvelope method), 10
getRelationshipType() (wsdiscovery.envelope.SoapEnvelope method), 10
getReplyTo() (wsdiscovery.envelope.SoapEnvelope
method), 10
getScheme() (wsdiscovery.uri.URI method), 14
getScopes() (wsdiscovery.envelope.SoapEnvelope
method), 10

[getScopes\(\)](#) (*wsdiscovery.service.Service method*), [13](#)
[getSequenceId\(\)](#) (*wsdiscovery.envelope.SoapEnvelope method*), [10](#)
[getTo\(\)](#) (*wsdiscovery.envelope.SoapEnvelope method*), [10](#)
[getTypes\(\)](#) (*wsdiscovery.envelope.SoapEnvelope method*), [10](#)
[getTypes\(\)](#) (*wsdiscovery.service.Service method*), [13](#)
[getValue\(\)](#) (*wsdiscovery.scope.Scope method*), [13](#)
[getXAddrs\(\)](#) (*wsdiscovery.envelope.SoapEnvelope method*), [10](#)
[getXAddrs\(\)](#) (*wsdiscovery.service.Service method*), [13](#)

H

[Hello](#), [3](#)

I

[incrementMessageNumber\(\)](#) (*wsdiscovery.service.Service method*), [13](#)
[isFinished\(\)](#) (*wsdiscovery.udp.UDPMessage method*), [14](#)

J

[join\(\)](#) (*wsdiscovery.threaded.NetworkingThread method*), [9](#)

M

[module](#)
[wsdiscovery.actions.bye](#), [11](#)
[wsdiscovery.actions.hello](#), [11](#)
[wsdiscovery.actions.probe](#), [12](#)
[wsdiscovery.actions.probematch](#), [12](#)
[wsdiscovery.actions.resolve](#), [12](#)
[wsdiscovery.actions.resolvematch](#), [12](#)
[wsdiscovery.daemon](#), [8](#)
[wsdiscovery.envelope](#), [10](#)
[wsdiscovery.message](#), [11](#)
[wsdiscovery.namespaces](#), [15](#)
[wsdiscovery.qname](#), [14](#)
[wsdiscovery.scope](#), [13](#)
[wsdiscovery.service](#), [13](#)
[wsdiscovery.threaded](#), [8](#)
[wsdiscovery.udp](#), [14](#)
[wsdiscovery.uri](#), [14](#)
[msgType\(\)](#) (*wsdiscovery.udp.UDPMessage method*), [14](#)
[MULTICAST](#) (*wsdiscovery.udp.UDPMessage attribute*), [14](#)

N

[NetworkingThread](#) (*class in wsdiscovery.threaded*), [8](#)

[NS_ACTION_BYE](#) (*in module wsdiscovery.namespaces*), [15](#)
[NS_ACTION_HELLO](#) (*in module wsdiscovery.namespaces*), [15](#)
[NS_ACTION_PROBE](#) (*in module wsdiscovery.namespaces*), [15](#)
[NS_ACTION_PROBE_MATCH](#) (*in module wsdiscovery.namespaces*), [15](#)
[NS_ACTION_RESOLVE](#) (*in module wsdiscovery.namespaces*), [15](#)
[NS_ACTION_RESOLVE_MATCH](#) (*in module wsdiscovery.namespaces*), [15](#)
[NS_ADDRESSING](#) (*in module wsdiscovery.namespaces*), [15](#)
[NS_DISCOVERY](#) (*in module wsdiscovery.namespaces*), [15](#)
[NS_SOAPENV](#) (*in module wsdiscovery.namespaces*), [15](#)

P

[parseByeMessage\(\)](#) (*in module wsdiscovery.actions.bye*), [11](#)
[parseHelloMessage\(\)](#) (*in module wsdiscovery.actions.hello*), [11](#)
[parseProbeMatchMessage\(\)](#) (*in module wsdiscovery.actions.probematch*), [12](#)
[parseProbeMessage\(\)](#) (*in module wsdiscovery.actions.probe*), [12](#)
[parseResolveMatchMessage\(\)](#) (*in module wsdiscovery.actions.resolvematch*), [12](#)
[parseResolveMessage\(\)](#) (*in module wsdiscovery.actions.resolve*), [12](#)
[parseSOAPMessage\(\)](#) (*in module wsdiscovery.message*), [11](#)
[Probe](#), [3](#)
[Probe match](#), [3](#)
[publishService\(\)](#) (*wsdiscovery.publishing.ThreadedWSPublishing method*), [7](#)

Q

[QName](#), [3](#)
[QName](#) (*class in wsdiscovery.qname*), [14](#)

R

[refresh\(\)](#) (*wsdiscovery.udp.UDPMessage method*), [14](#)
[removeSourceAddr\(\)](#) (*wsdiscovery.threaded.NetworkingThread method*), [9](#)
[removeSourceAddr\(\)](#) (*wsdiscovery.threaded.ThreadedNetworking method*), [9](#)
[Resolve](#), [3](#)
[Resolve match](#), [3](#)

```
run () (wsdiscovery.threaded.AddressMonitorThread
method), 8
run () (wsdiscovery.threaded.NetworkingThread
method), 9
```

S

```
schedule_stop () (wsdiscovery.threaded._StoppableDaemonThread
method), 10
Scope, 3
Scope (class in wsdiscovery.scope), 13
searchServices () (wsdiscovery.discovery.ThreadedWSDiscovery
method), 7
sendMulticastMessage () (wsdiscovery.threaded.ThreadedNetworking
method), 9
sendUnicastMessage () (wsdiscovery.threaded.ThreadedNetworking
method), 10
Service (class in wsdiscovery.service), 13
setAction () (wsdiscovery.envelope.SoapEnvelope
method), 10
setEPR () (wsdiscovery.envelope.SoapEnvelope
method), 10
setEPR () (wsdiscovery.service.Service method), 13
setInstanceId () (wsdiscovery.envelope.SoapEnvelope method), 10
setInstanceId () (wsdiscovery.service.Service
method), 13
setMessageId () (wsdiscovery.envelope.SoapEnvelope method), 10
setMessageNumber () (wsdiscovery.envelope.SoapEnvelope method), 11
setMessageNumber () (wsdiscovery.service.Service
method), 13
setMetadataVersion () (wsdiscovery.envelope.SoapEnvelope method), 11
setMetadataVersion () (wsdiscovery.service.Service method), 13
setProbeResolveMatches () (wsdiscovery.envelope.SoapEnvelope method), 11
setRelatesTo () (wsdiscovery.envelope.SoapEnvelope method), 11
setRelationshipType () (wsdiscovery.envelope.SoapEnvelope method), 11
setRemoteServiceByeCallback () (wsdiscovery.discovery.ThreadedWSDiscovery
method), 7
setRemoteServiceHelloCallback () (wsdiscovery.discovery.ThreadedWSDiscovery
method), 7
setReplyTo () (wsdiscovery.envelope.SoapEnvelope
method), 11
```

```
setScopes () (wsdiscovery.envelope.SoapEnvelope
method), 11
setScopes () (wsdiscovery.service.Service method), 13
setSequenceId () (wsdiscovery.envelope.SoapEnvelope method), 11
setTo () (wsdiscovery.envelope.SoapEnvelope method), 11
setTypes () (wsdiscovery.envelope.SoapEnvelope
method), 11
setTypes () (wsdiscovery.service.Service method), 13
setXAddrs () (wsdiscovery.envelope.SoapEnvelope
method), 11
setXAddrs () (wsdiscovery.service.Service method), 13
SoapEnvelope (class in wsdiscovery.envelope), 10
start () (wsdiscovery.discovery.ThreadedWSDiscovery
method), 8
start () (wsdiscovery.publishing.ThreadedWSPublishing
method), 7
start () (wsdiscovery.threaded.NetworkingThread
method), 9
start () (wsdiscovery.threaded.ThreadedNetworking
method), 10
stop () (wsdiscovery.discovery.ThreadedWSDiscovery
method), 8
stop () (wsdiscovery.publishing.ThreadedWSPublishing
method), 7
stop () (wsdiscovery.threaded.ThreadedNetworking
method), 10
```

T

```
ThreadedNetworking (class in wsdiscovery.threaded), 9
ThreadedWSDiscovery (class in wsdiscovery.discovery), 7
ThreadedWSPublishing (class in wsdiscovery.publishing), 7
```

U

```
UDPMessage (class in wsdiscovery.udp), 14
UNICAST (wsdiscovery.udp.UDPMessage attribute), 14
URI (class in wsdiscovery.uri), 14
```

W

```
wsdiscover command line option
--address <address>, 5
--capture <capture>, 5
--loglevel <loglevel>, 5
--port <port>, 5
--scope <scope>, 5
-a, 5
-c, 5
-l, 5
```

```

    -p, 5
    -s, 5
wsdiscovery.actions.bye
    module, 11
wsdiscovery.actions.hello
    module, 11
wsdiscovery.actions.probe
    module, 12
wsdiscovery.actions.probematch
    module, 12
wsdiscovery.actions.resolve
    module, 12
wsdiscovery.actions.resolvematch
    module, 12
wsdiscovery.daemon
    module, 8
wsdiscovery.envelope
    module, 10
wsdiscovery.message
    module, 11
wsdiscovery.namespaces
    module, 15
wsdiscovery.qname
    module, 14
wsdiscovery.scope
    module, 13
wsdiscovery.service
    module, 13
wsdiscovery.threaded
    module, 8
wsdiscovery.udp
    module, 14
wsdiscovery.uri
    module, 14
wspublish command line option
    --address <address>, 6
    --capture <capture>, 6
    --loglevel <loglevel>, 6
    --port <port>, 6
    --scope <scope>, 6
    --typename <typename>, 6
    -a, 6
    -c, 6
    -l, 6
    -p, 6
    -s, 6
    -t, 6

```