

# SOAP Web Services Developer Guide

Clearspace exposes a lot of its functionality as web services. This topic describes the technology on which those services are based and provides examples on how you can use the services.

- [Overview](#) (page 1)
- [Web Service Style](#) (page 1)
- [Authentication](#) (page 1)
- [Objects](#) (page 2)
- [Services](#) (page 2)
- [Client API Examples](#) (page 3)
- [Client API Dependencies](#) (page 5)

## Overview

You can get Clearspace WSDL description files by requesting the following URL from your Clearspace instance: `http://<hostname>:<portnumber>/<context>/rpc/soap/`.

**Note:** By default, web services are disabled in Clearspace. You can enable them in the admin console. In the console, go to **System > Settings > Web Services**, then click **Enable** for the style you want. Be sure that the **User Access** and **Force SSL** settings are what you want also.

## Web Service Style

Clearspace Web Services use Document-Literal style binding with wrapped parameter format. This style was chosen for a number of reasons: it is WS-I compliant and the message is easily validated. There is also a general push towards Document-Literal style web services.

For a general description about web services styles see: <http://www.ibm.com/developerworks/webservices/library/ws-whichwsdl/>

## Authentication

Clearspace Web Services uses the [Username Token Profile V1.0](#) (pdf) specification. The Username Token Profile specification is part of the [OASIS Web Services Security \(WS-Security\)](#) specification.

A header containing the username and password must be passed into every request. Here's an example:

```
<wsdl:Envelope xmlns:soap="..." xmlns:wsse="..." >
  <wsdl:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>admin</wsse:Username>
      </wsse:UsernameToken>
    </wsse:Security>
  </wsdl:Header>
</wsdl:Envelope>
```

```

    <wsse:Password>password</wsse:Password>
  </wsse:UsernameToken>
</wsse:Security>
</wsdl:Header>
</wsdl:Envelope>

```

- The namespace for WSDL prefix is <http://schemas.xmlsoap.org/wsdl/>
- The namespace for WSSE prefix is <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd>

## Web Services Client API

The following tables list the main classes of the API you use to access Clearspace web services. You'll find references for these in the [Clearspace web services Javadoc](#).

### Objects

Object	Description
WSCommunity	A container for threads and a hierarchy of other communities.
WSDocument	Encapsulates a document in the Clearspace system.
WSForumMessage	Encapsulates discussion message data.
WSForumThread	A container for a hierarchy of ForumMessages.
WSGroup	Organizes users into a group for easier permissions management.
WSJiveObject	Super-class of other web service objects.
WSUser	Provides information about and services for users of the system.
WSWatch	A way for a user to track updates to an object.

### Services

Service	Description
AddressBookService	Provides ability to interact with the private message addressbook.

AttachmentService	A web service for managing attachment settings.
CommunityService	Provides the ability to manipulate communities.
DocumentService	Provides methods to load and manipulate documents.
ForumService	Provides the ability to manipulate discussion messages.
GroupService	Provides the ability for managing groups and group membership.
PermissionService	Provides a web service for managing permissions on users and groups.
PrivateMessageService	Provides the ability to manipulate private messages.
SearchService	Provides the ability to search for content.
SystemPropertiesService	Provides a web service for managing system properties.
UserService	Provides a web service for managing users, avatars, and status levels.
WatchService	A service for manipulating a user's watches on objects.

## API Examples

Jive Software provides a Java client API for SOAP-based web services. The entry point to the client API is the class `com.jivesoftware.community.webservices.ServiceLocator`. The `ServiceLocator` handles authentication and provides getters for all of the services.

The following examples show how to use the client APIs to connect to the Clearspace service and perform various tasks. Note that most connections use the HTTPS protocol. We recommend this approach because it's the most secure. The system can be configured to use plain HTTP connections.

### Acquire the Root Community

```
ServiceLocator locator = new ServiceLocator("https://myclearspace.com", "admin", "password");
CommunityService communityService = locator.getCommunityService();
Community community = communityService.getCommunity(1);
```

### Create a New Thread

```
ServiceLocator locator = new ServiceLocator("https://myclearspace.com", "admin", "password");
CommunityService communityService = locator.getCommunityService();
```

```

ForumService forumService = locator.getForumService();
UserService userService = locator.getUserService();

// We want this thread to appear in the community with ID 33
Community community = communityService.getCommunity(33);

// Get the user we want to post the message as
User user = userService.getUser("myUsername");

// Create the thread
ForumThread thread = forumService.createThread("My Test Message", "My message body",
    community.getID(), user.getID());

```

## Add a Reply to a Message

```

// The message to reply to
long messageID = 1221;
// The user we're posting the message as
long userID = 12123;

ServiceLocator locator = new ServiceLocator("https://myclearspace.com", "admin", "password");
ForumService contentService = locator.getContentService();
// Create the reply
ForumMessage reply = forumService.createReplyMessage("my subject", "my response",
    message, userID);

```

## Add an Attachment to a Message

```

// The message we're going add an attachment to
long messageID = 12123;

// Create a FileDataSource for the attachment. Do this with a DataSource object
// (part of the Java Activation Framework)
javax.activation.DataSource myData =
    new javax.activation.FileDataSource("/home/me/myfile.doc");

ServiceLocator locator = new ServiceLocator("https://myclearspace.com", "admin",
    "password");
ForumService svc = locator.getForumService();
svc.addAttachmenToMessage("myfile", myData, messageID);

```

## Create a New User

```

ServiceLocator locator = new ServiceLocator("https://myfor.com", "admin",
    "password");
UserService svc = locator.getUserService();

User user = svc.createUser("johnsmith", "password", "johns@foo.com");

```

## Get First 1000 Messages in a Thread

```
ServiceLocator locator = new ServiceLocator("https://myclearspace.com", "admin",
    "password");
ForumService forumService = locator.GetForumService();

long myThreadID = 333L;
ArrayList<ForumMessage> messages = new ArrayList<ForumMessage>();
long[] messageIDs = forumService.getMessageIDsByThreadID(myThreadID);

for (long messageID : messageIDs) {
    ForumMessage current = messageService.getForumMessage(messageID);
    messages.add(current);
}
```

## Execute a Search

```
ServiceLocator locator = new ServiceLocator("https://myclearspace.com", "admin",
    "password");
SearchService service = locator.getSearchService();

Query query = new Query();
query.setQueryString("database java");
// Anything later than Jan 1, 2006
query.setAfterDate(new GregorianCalendar(2006, 1, 1).getTime());
query.setSortField(Query.DATE);
query.setSortOrder(Query.ASCENDING);

// Only grab the first 100 entries
long[] messageIDs = service.messageSearch(query, 1, 100);
```

## Grant a Specific Group Read Permission to a Specific Community

```
long myCommunityID = 3333L;

ServiceLocator locator = new ServiceLocator("https://myclearspace.com", "admin", "password");
GroupService groupService = locator.getGroupService();
PermissionService permissionService = locator.getPermissionService();

Group group = groupService.getGroup("myGroup");
permissionService.addCommunityPermissionToGroup(com.jivesoftware.clearspace.Permissions.READ_COMMUNITY,
    true, group.getID(), myCommunityID);
```

## Client API Dependencies

The client APIs have the following dependencies. These are included with Clearspace.

Library File	Description
activation.jar	Sun Java Activation Framework
commons-codec.jar	Apache Commons Codec library
commons-httpclient.jar	Apache Commons HTTP Client library
commons-logging.jar	Apache Commons Logging, A generic logging interface
cxf.jar	CXF libraries
geronimo.jar	Apache Geronimo, a J2EE server
google-collect.jar	Google utility classes for working with collections.
javamail.jar	Sun Mail Framework, used for attachments
jaxb-api.jar	Sun JAXB API
jaxb-impl.jar	Sun JAXB Impl
jaxb-xjc	Sun JAXB XJC
jaxen.jar	Jaxen, a XPath engine
jdom.jar	JDom, a library for manipulating XML as simple Java objects
jive-license.jar	Jive licensing internal API
stax-utils.jar	Utils for stax
sun-jaxws-api.jar	Sun Jax web services API
sun-saaj-api.jar	Sun SOAP Attachments API
sun-saaj-impl.jar	The Sun implementation of SAAJ
wSDL4j.jar	An API for manipulating WSDLs
wss4j.jar	The Apache implementation of the WS-Security spec
wstx-asl.jar	Woodstox is a STAX implementation from Codehaus
xml-resolver.jar	Required by CXF
XmlSchema.jar	XML Schema support
xmlsec.jar	XML Security