

# Clearspace 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.

## Contents

- [Overview](#)
- [Web Service Style](#)
- [Authentication](#)
- [Objects](#)
- [Services](#)
- [Client API Examples](#)
- [Client API Dependencies](#)

## Overview

Jive Clearspace Web Services implement the [SOAP protocol](#). WSDL description files are available as part of the distribution or by requesting the following URL from your Jive Clearspace instance:  
<http://myhost.com/clearspace/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

Jive 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, the message is easily validated, and it works well with .NET web services. 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

Jive 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.

```

<wsdl:Envelope xmlns:soap="..." xmlns:wssse="..." >
  <wsdl:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>admin</wsse:Username>
        <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
Community	A container for threads and a hierarchy of other communities.
Document	A Document object encapsulates a document in the Clearspace system.
ForumMessage	A ForumMessage encapsulates message data.
ForumThread	A ForumThread is a container for a hierarchy of ForumMessages.
Group	Organizes users into a group for easier permissions management.
JiveObject	Super-class of other jive webservice objects.
User	Provides information about and services for users of the system.
Watch	A watch is a way for a user to track updates to an object.

### Services

Service	Description
AddressBookService	Provides ability to interace with the Private Message addressbook.
AttachmentManagerService	A web service for managing attachment settings.
CommunityService	Provides the ability to manipulate communities.

DocumentService	This service provides methods to load and manipulate documents.
ForumService	Provides the ability to manipulate forum messages.
GroupService	Provides a the ability for managing groups and group membership.
PermissionService	Provides a webservice 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 Jive System Properties.
UserService	Provides a webservice for managing user's, avatar's, and status levels.
WatchService	A service for manipulating a user's watches on objects.

## API Examples

Jive Software provides a Java and .NET client APIs. For both languages, 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.

Below are a few example of how to use the client APIs to connect to the Jive Clearspace service and perform various tasks. Note, most connections use the HTTPS protocol. We recommend this approach as it's the most secure. The system can be configured to use plain HTTP connections. For descriptions of the various service objects below, see the \#Services.

### Acquire the Root Community

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

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

```

```

ServiceLocator locator = new ServiceLocator("https://myclearspace.com", "admin", "password");
CommunityService communityService = locator.communityService;
ForumService forumService = locator.ForumService;
UserService userService = locator.UserService;

// 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.ID, user.ID);

```

### 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);

```

```

// 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 forumService = locator.ContentService;
// 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);

```

```
// The message we're going add an attachment to
long messageID = 12123;
bytes[] myData = File.ReadAllBytes("c:\myfile.txt");

ServiceLocator locator = new ServiceLocator("https://myclearspace.com", "admin",
    "password");
ForumMessageService svc = locator.ForumMessageService;
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");
```

```
ServiceLocator locator = new ServiceLocator("https://myclearspace.com", "admin",
    "password");
UserService svc = locator.userService;

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);
}
```

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

long myThreadID = 333L;
List<ForumMessage> messages = new List<ForumMessage>();
long[] messageIDs = svc.GetMessageIDsByThreadID(myThreadID);

for (long messageID in messageIDs) {
    ForumMessage current = svc.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);
```

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

Query query = new Query();
query.queryString = "database java";
query.AfterDate = new DateTime(2006, 1, 1); // Anything later than Jan 1, 2006
query.SortField = 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);
```

```
long myCommunityID = 3333L;

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

Group group = groupService.GetGroup("myGroup");
permissionService.AddCommunityPermissionToGroup(Permissions.READ_COMMUNITY, true, group.ID,
    myCommunityID);
```

## Client API Dependencies

The client APIs have the following dependencies, which are included in the distributions.

### The Java client API dependencies.

Jive Webservice libraries depend on the following

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.
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 mainulating xml as simple java objects.
stax-api.jar	API for XML pull style parsing.
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
wss4j.jar	The Apache implementation of the WS-Security spec.
wstx-asl.jar	Woodstox a fast STAX implementation from Codehaus.
xfire-all.jar	XFire libraries
xfire-jsr181-api.jar	XFire java 1.5 annotations.
XmlSchema.jar	XML Schema support.
xmlsec.jar	XML Security.

**The .NET client API dependencies.**

<b>Library File</b>	<b>Description</b>
Microsoft.Web.Services3.dll	Microsoft web services extensions, used for WS-Security support.