

# Using ADOBE® CONNECT™ 8 Web Services



© 2010 Adobe Systems Incorporated. All rights reserved.

Using Adobe® Connect™ 8 Web Services

This user guide is protected under copyright law, furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this guide.

This user guide is licensed for use under the terms of the Creative Commons Attribution Non-Commercial 3.0 License. This License allows users to copy, distribute, and transmit the user guide for noncommercial purposes only so long as (1) proper attribution to Adobe is given as the owner of the user guide; and (2) any reuse or distribution of the user guide contains a notice that use of the user guide is governed by these terms. The best way to provide notice is to include the following link. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/>

Adobe, the Adobe logo, Acrobat Connect, Adobe Captivate, Adobe Connect, Authorware, Flash, Flash Builder, Flex, and Flex Builder are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. All other trademarks are the property of their respective owners.

Updated Information/Additional Third Party Code Information available at [www.adobe.com/go/thirdparty/](http://www.adobe.com/go/thirdparty/)

Portions include software under the following terms:

This software is based in part on the work of the Independent JPEG Group.

Flash 9 video compression and decompression is powered by On2 TrueMotion video technology. © 1992-2005 On2 Technologies, Inc.

Sorenson Spark(tm) video compression and decompression technology licensed from Sorenson Media, Inc.

MPEG Layer-3 audio coding technology licensed from Fraunhofer IIS and Thomson.

RealDuplex™ Acoustic Echo Cancellation is Copyright © 1995-2004 SPIRIT.

This product contains either BSAFE and/or TIPEM software by RSA Security, Inc.

This product includes software developed by the Apache Software Foundation ([www.apache.org/](http://www.apache.org/)).

Portions © 1995-2005 Jean-loup Gailly and Mark Adler. This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software. Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions: 1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required. 2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software. 3. This notice may not be removed or altered from any source distribution. Jean-loup Gailly ([jloup@gzip.org](mailto:jloup@gzip.org)) Mark Adler ([madler@alumni.caltech.edu](mailto:madler@alumni.caltech.edu))

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Notice to U.S. Government End Users. The Software and Documentation are "Commercial Items," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

# Contents

## Chapter 1: Before you begin

Development environment .....	1
Additional resources .....	1
Conventions .....	2

## Chapter 2: Architecture

Data flow .....	3
Making your first API call .....	6

## Chapter 3: Login and requests

Log in from an application .....	10
Send a request in an XML document .....	15
Parse a response with XPath .....	15
Parse an error response .....	16
Log a user out .....	17

## Chapter 4: Basics

Find a principal-id .....	18
List principals or guests .....	19
Create users .....	20
Update users .....	21
Create custom fields .....	22
Create groups .....	22
Find SCOs .....	24
Download files .....	27
Check permissions .....	28

## Chapter 5: Meetings

Using web services with Adobe Connect meetings .....	30
Find meetings .....	30
Display meetings .....	31
Create meeting room URLs .....	32
Create meetings .....	33
Set or reset a meeting passcode .....	35
Create customized meetings .....	36
Invite users to meetings .....	37
Remove users from meetings .....	39
Calculate meeting usage .....	40
Check meeting quotas .....	41
Get meeting archives .....	41
Get meeting poll results .....	42
Launch meetings with external authentication .....	43
Configure compliance settings .....	44

**Contents**

**Chapter 6: Training**

Using web services with Adobe Connect Training ..... 47

Training library permissions ..... 47

Find courses and curriculums ..... 48

Create a course ..... 50

View a user’s training ..... 50

Enroll one user ..... 51

Enroll a large number of users ..... 53

View curriculum information ..... 55

Report scores ..... 56

**Chapter 7: Action reference**

What’s new in Adobe Connect 8 Web Services ..... 58

Sample action ..... 58

Actions ..... 59

**Chapter 8: Filter and sort reference**

filter-definition ..... 214

sort-definition ..... 215

**Chapter 9: Using the Telephony XML API**

telephony-profile-delete ..... 217

telephony-profile-info ..... 218

telephony-profile-list ..... 219

telephony-profile-update ..... 220

telephony-provider-conf-number-update ..... 221

telephony-provider-delete ..... 222

telephony-provider-dial-in-info-update ..... 223

telephony-provider-field-delete ..... 223

telephony-provider-field-list ..... 224

telephony-provider-field-update ..... 226

telephony-provider-info ..... 228

telephony-provider-list ..... 231

telephony-provider-update ..... 232

**Chapter 10: Common reference**

Common XML elements and attributes ..... 234

**Chapter 11: Sample application**

Getting started with the sample application ..... 248

Build an adapter class ..... 248

Log the user in ..... 249

Send XML requests ..... 251

Parse XML responses ..... 252

Display user information ..... 253

List a user’s meetings ..... 254

Create and update meetings ..... 255

Display meeting detail ..... 257

# Chapter 1: Before you begin

Adobe® Connect® exposes web services that clients can call to exchange data with Adobe Connect accounts. You can use web services with Adobe Connect hosted accounts and with accounts on Adobe Connect licensed servers.

This guide explains how an application calls Adobe Connect web services and interprets the XML response. It is intended for developers who want to build custom applications for Adobe Connect or integrate it with another system such as a learning management system or LDAP directory service.

Before you use this guide, you should understand the basics of XML and of using HTTP to communicate with a server from a client application. This guide includes some Java™ code samples, but it does not presume that you are using one specific language or environment.

## Development environment

Adobe Connect Web Services allows you to use any language or platform that can send and receive XML over HTTP to develop custom applications. For example, you can use Java and the J2EE platform, C#.NET, PHP, a portal server, or any web development platform. Most custom applications are web applications or portals.

In general, you may find these types of tools useful:

- An XML parser code library, if your programming language supports XML parsing.
- A cookie management code library, to help you manage the session cookies Adobe Connect returns.
- A tool for viewing HTTP request and response headers in a browser. Many such tools are available on the Internet.

## Additional resources

You can find many useful resources on the Internet that provide information about Adobe Connect, web services and XML, and other technologies that Adobe Connect uses.

### Adobe Connect

**Adobe Connect User Community** The Adobe Connect User Community at [connectusers.com](http://connectusers.com) is the hub of the Adobe Connect community. This site has forums, tutorials, events, announcements, a partner showcase and much more.

**Adobe Connect Help Support Center** The Adobe Connect [Help and Support Center](#) contains the Adobe Connect documentation and Support contact information.

### XML and web services

**The Web Services Primer** at the Xml.com website ([xml.com](http://xml.com)) is a good introduction to web services.

**The XML Tutorial** at the W3Schools website ([w3schools.com](http://w3schools.com)) can help you get started with XML.

**The XPath Tutorial** also at the W3Schools website ([w3schools.com](http://w3schools.com)), describes XPath, which parses an XML document so that you can use it in an application.

**Before you begin**

**The XSLT Tutorial**, a third tutorial at the W3Schools website (w3schools.com), teaches you XSL Transformations, which you use to convert XML data to other formats.

**The XSL Transformations (XSLT) specification** at the W3C website (w3.org) is the official definition of XSLT, from the standards committee who created it.

**Numeric Representation of Dates and Time**, at the International Organization for Standardization website (iso.org), provides information about how to use the ISO 8601 standard date and time format.

**Date and Time Formats** at the W3C website (w3.org) is the official definition of the ISO 8601 date and time format.

## Other technologies

**Flash Player Developer Center** and Flash Media Server Developer Center, both available from the [Adobe Developer Center](#), offer articles, samples, and insights to developing applications that use Adobe Flash Player and Adobe Flash Media Server.

**SCORM Concepts**, at the Eduworks Corporation website (eduworks.com), is a tutorial about the Shareable Content Object Reference Model and describes Shareable Content Objects (SCOs) and Learning Management Systems (LMSs).

**An LDAP Roadmap** at the Kings Mountain Systems website (www.kingsmountain.com), provides a useful overview of the Lightweight Directory Access Protocol (LDAP). This site might provide good background material or links for developers integrating an LDAP directory with Adobe Connect.

**Microsoft SQL Server** Adobe Connect uses a Microsoft SQL Server database, which your custom applications retrieve data from and write data to. You may find useful resources at the Microsoft SQL Server Developer Center (msdn.microsoft.com) including references, community, support, and other information.

## Conventions

This guide uses industry standard conventions for displaying code that you are already familiar with.

However, API reference is a formal definition of the API contract between a calling application and the server. As such, the syntax definitions of request URLs should be described.

We have placed distinct sections of a request URL on separate lines for readability, like this:

```
http://server_name/api/xml
    ?action=custom-fields
      &filter-definition=value
        &session=BreezeSessionCookieValue
```

When you enter a request URL in the address bar of a browser or construct it in an application, enter it or construct it as a single line:

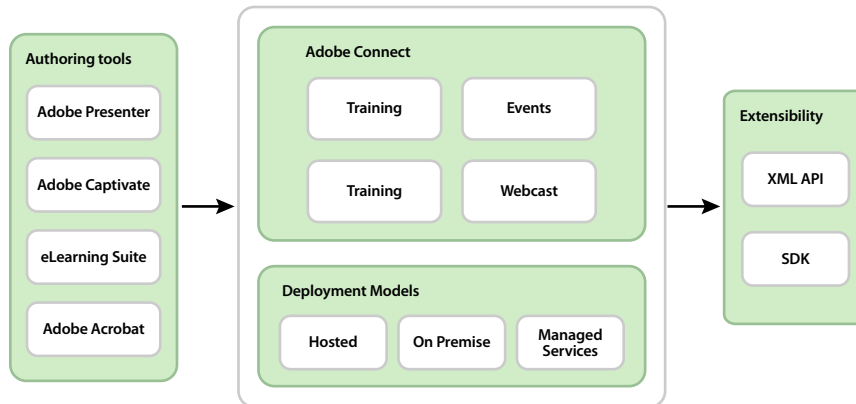
```
https://example.com/api/xml?action=custom-fields&filter-name=location
```

Syntax elements in blue code font represent definitions that you construct, with a hyperlink to the syntax of the definition.

## Chapter 2: Architecture

Adobe® Connect™ Web Services is the web service layer over the Adobe Connect Server suite of applications.

Web services allow you to build portals or web applications that integrate Adobe Connect functionality and reporting information with third-party systems such as portals, customer relationship management systems, and enterprise resource planning systems.



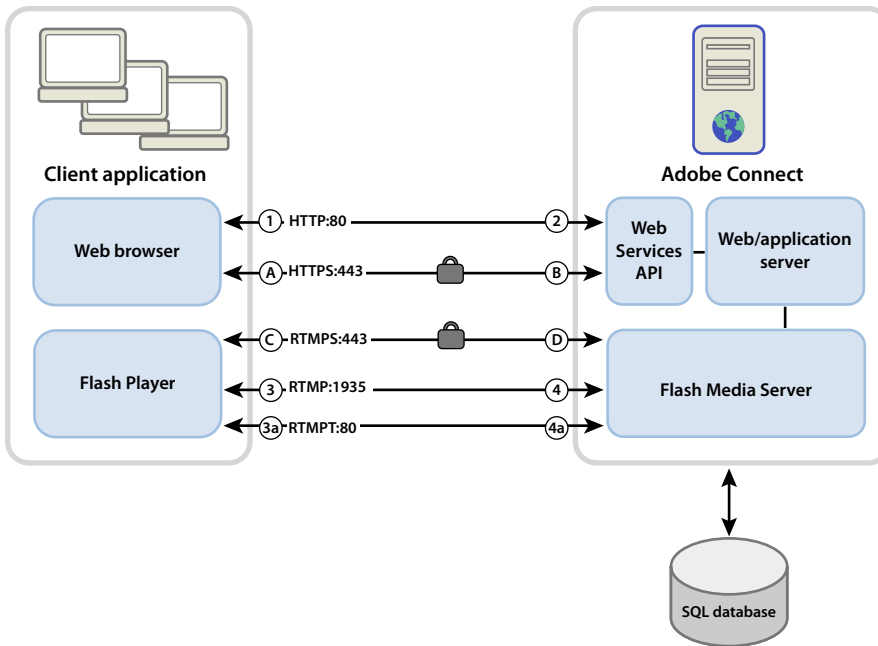
*Adobe Connect Web Services provides meeting, training, and events functionality to your applications through its XML API.*

As an example, you might have a central user management system, such as an LDAP directory, Microsoft Active Directory, or another third-party system, that is an integral part of your business processes.

Using web services, you can write an application that synchronizes users between your system and Adobe Connect. The application can use the J2EE platform or another technology of your choice to pull a list of users from the directory, compare it against a list of Adobe Connect users, and then perform requested updates within the Adobe Connect user repository, such as adding or deleting users or groups.

### Data flow

The data flows between client applications and Adobe Connect are shown in the following diagram. Custom applications that you write use paths 1 to 2 and A to B. Adobe Connect applications (such as Adobe Connect Meeting, Adobe Connect Training, or Adobe Connect Events) can use any of the data flow paths.



The data flow between Adobe Connect and client applications

The data flow can be encrypted with SSL or unencrypted.

**Unencrypted** If the data flow is unencrypted, connections are made over HTTP and Adobe Real Time Messaging Protocol (RTMP) and follow the paths described in the following table.

Diagram number	Description
1	The client web browser requests an Adobe Connect meeting or content URL over port HTTP:80 (connection paths may vary).
2	The web server responds with content transfer or provides the client browser with information to enter Adobe Connect.
3	Adobe Flash® Player requests a connection to Adobe Flash Media Server over RTMP:1935 and HTTP:80.
4	Flash Media Server responds, and a persistent connection is opened to stream meeting traffic to the browser.
3a (alternate)	In some cases, Flash Player requests a connection to the Flash Media Server, but can only obtain a tunneled connection over RTMPT:80.
4a (alternate)	Flash Media Server responds, and a tunneled connection is opened to stream meeting traffic to the browser.

**Encrypted** If the data flow is encrypted, connections are made securely over HTTPS and RTMPS (Real Time Messaging Protocol over SSL), as follows.

Diagram number	Description
A	The client web browser requests a secure meeting or content URL over an encrypted connection on HTTPS:443 (connection paths may vary).
B	The web/application server responds with an encrypted content transfer or provides the client with information to make an encrypted connection to Adobe Connect.
C	Flash Player requests an encrypted connection to Flash Media Server over RTMPS:443.
D	Flash Media Server responds, and a persistent connection is opened to stream meeting traffic to the browser.



## Custom applications

Adobe Connect Web Services provides an XML API, so your application must be able to communicate with Adobe Connect using XML over HTTP or XML over HTTPS. Your application calls the API by building a request URL and passing it one or more parameters, either as name/value pairs or as an XML document. Web Services returns an XML response, from which you can extract values.

Custom applications retrieve metadata from the Adobe Connect database. Metadata includes meeting or course names and times, meeting room URLs, content URLs, and report information.

The data flow for a custom application retrieving metadata from the database is from a client web browser, to the client web application server, to the XML API, the Adobe Connect web application server, and the SQL database—and then back again.

The data flow between a custom application and Adobe Connect works like this:

- 1 A user accesses your custom application from a web browser.
- 2 The application calls the XML API over HTTP:80 or HTTPS:443.
- 3 The Adobe Connect web application server authorizes the application and its users, retrieves metadata from the SQL database, and returns the metadata.
- 4 On the client side, your web or application server, XML parser, and software libraries handle the response and return it to your application.
- 5 The user continues to work in your custom application, and clicks a meeting or content URL. At this point, the user accesses a Adobe Connect application to enter a meeting room, and the typical data flow between a Adobe Connect application and the server begins.

## Adobe Connect applications

Adobe Connect applications call the server using the same Web Services XML API that you use from a custom application.

In general, content is transported over HTTP port 80 or HTTPS port 443. Content includes slides, HTTP pages, SWF files, and files transferred through the FileShare pod. These are default port numbers that you can configure (see *Migrating, Installing, and Configuring Adobe Connect Server* for details).

Streamed, real-time communications from Flash Media Server are transported over RTMP port 1935. Streamed communications include audio, video (webcam and FLV), file share, and chat. Meeting state is also maintained over RTMP port 1935.

## Components of Adobe Connect

Adobe Connect is architected with two server components, and each server uses a SQL database.

**The web application server** The web application server is the brains of Adobe Connect. It contains and executes all of the business logic needed to deliver content to users. It handles access control, security, quotas, and licensing, as well as management functions such as clustering, failover, and replication.

The web application server also handles Adobe Connect Central, the application through which you view and manage your organization's content and users—when you are not using a custom application or integrated third-party system. The metadata describing content and users can be stored in either single or multiple replicated SQL databases. The web application server is stateless, which means that scaling is near linear.

**Flash Media Server** Flash Media Server is the muscle of Adobe Connect. Flash Media Server streams audio, video, and rich media content using RTMP. When a meeting is recorded and played back, audio and video are synchronized, or content is converted and packaged for real-time screen sharing, Flash Media Server does the job.

Flash Media Server also plays a vital role in reducing server load by caching frequently accessed web pages, streams, and shared data.

**The SQL database** Adobe Connect uses the Microsoft SQL Server database for persistent storage of transactional and application metadata, including users, groups, content, and reporting information. The XML API retrieves metadata stored in the database. The database can be implemented with either the Microsoft SQL Server Desktop Engine (MSDE) or the full version of Microsoft SQL Server 2005.

## Making your first API call

Adobe Connect Web Services uses a servlet framework to handle XML API requests. In the data flow diagram, the servlet framework is represented by the API component. The API servlet receives XML requests from clients and returns XML responses from the web application server and the database.

A request to the XML API is formatted as an HTTP request URL that the API servlet handles. A request URL has an action name and parameters in name/value pairs, like this:

```
https://example.com/api/xml?action=sco-info&sco-id=2006334909
```

If you have access to a Adobe Connect account in which you can test API calls, you can experiment. In fact, Adobe recommends testing API calls in the browser while you learn the API and write applications.

Before you begin, it's useful to install a tool that allows you to view HTTP request and response headers in your browser.

### Call common-info in a browser

- 1 (Optional) Enable a tool for viewing HTTP headers in your browser.
- 2 Open a browser and navigate to your Adobe Connect login page.
- 3 Without logging in, delete the part of the URL after the domain name and add a call to `common-info`:

```
https://example.com/api/xml?action=common-info
```

The response from `common-info` gives you information about your session with the server, especially the cookie that identifies your session:

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <common locale="en" time-zone-id="85">
    <cookie>breezbryf9ur23mbokzs8</cookie>
    <date>2008-03-13T01:21:13.190+00:00</date>
    <host>https://example.com</host>
    <local-host>abc123def789</local-host>
    <url>/api/xml?action=common-info</url>
    <version>connect_700_r641</version>
    <user-agent>
      Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1;
        .NET CLR 1.1.4322)
    </user-agent>
  </common>
</results>
```

When you log a user in from an application, you need to send the cookie value back to the server to identify the user's session (see [Log in from an application](#)).

### Call principal-list in a browser

Once you have the BREEZESESSION cookie value from `common-info`, the browser adds it to the request header on your next request.

- 1 In a web browser, log in to Adobe Connect. Change the browser URL to call `principal-list`:

```
https://example.com/api/xml?action=principal-list
```

- 2 Check the request header. This time it sends the BREEZESESSION cookie value back to the server:

```
GET /api/xml?action=principal-list HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)
Host: example.com
Connection: Keep-Alive
Cookie: BREEZESESSION=breezbryf9ur23mbokzs8
```

- 3 Check the response, which lists all principals on the server, each in its own `principal` element.

**Architecture**

```

<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <principal-list>
    <principal principal-id="624526" account-id="624520" type="user"
      has-children="false" is-primary="false" is-hidden="false">
      <name>joe harrison</name>
      <login>jharrison@example.com</login>
      <email>jharrison@example.com</email>
    </principal>
    <principal principal-id="624550" account-id="624520" type="user"
      has-children="false" is-primary="false" is-hidden="false">
      <name>bob jones</name>
      <login>bjones@example.com</login>
      <email>bjones@example.com</email>
    </principal>
    ...
  </principal-list>
</results>

```

**Add filters and sorts**

Many actions in the API allow you to add a filter to return only certain response elements or a sort to display response elements in a certain order.

A filter is a special parameter that starts with the keyword `filter`, followed by an optional modifier, then a field name and a value. These are all examples of filters:

- `filter-name=jazz doe` (which matches results with the exact name *jazz doe*)
- `filter-like-name=jazz` (which matches any results that contain *jazz* in the name)
- `filter-out-type=user` (which returns any results that do not have a type of *user*)

These are just a few filter types, and you can find more in [filter-definition](#). Check an action in the reference (at “[Action reference](#)” on page 58) to see whether its response can be filtered. In general, if an action allows filters, you can use them on any response element or attribute.

A sort is another special parameter that starts with the keyword `sort` (or `sort1` or `sort2`), followed by a field name and then one of the keywords `asc` or `desc`, for example:

- `sort-name=asc` (to sort in ascending order by name)
- `sort-group-id=desc` (to sort in descending order by `group-id`)

These are just a few sort examples. You can test sorts in the browser or see [sort-definition](#) for more.

**Make a call with a filter and sort**

1 Call `principal-list` again, displaying only groups and sorting them alphabetically by name:

```

https://example.com/api/xml?action=principal-list&filter-type=group
&sort-name=asc

```

2 To tighten the response, choose a group from the list and filter on its name:

```

https://example.com/api/xml?action=principal-list&filter-name=developers

```

This time, only one group is returned:

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <principal-list>
    <principal principal-id="2007105030" account-id="624520"
      type="group" has-children="true" is-primary="false"
      is-hidden="false">
      <name>developers</name>
      <login>developers</login>
    </principal>
  </principal-list>
</results>
```

## Where to go from here

At this point, you can continue to test calls in the browser and observe how they work. It's the best and easiest way to learn the XML API. When you need more information, turn to any of these sources:

- The API reference in [“Action reference”](#) on page 58
- [“Login and requests”](#) on page 10 for information on how to log users in from applications
- [“Basics”](#) on page 18 to learn the three basic concepts underlying the API
- [“Meetings”](#) on page 30 if you want to create and manage meetings from an application
- [“Training”](#) on page 47 if you are building a training application

# Chapter 3: Login and requests

This chapter explains how to log a user in from your application, make requests, handle responses, and log the user out.

There are several ways to accomplish most of these tasks, depending on your development environment, server configuration, and application design.

## Log in from an application

Any custom application you write that uses Adobe® Connect™ Web Services functionality or integrates with a third-party system needs to log in a user to Adobe Connect. In its simplest form, the process of logging in calls the `login` action.

However, the technique for logging in varies according to whether you use cookie management, have a licensed server or a hosted account, and authenticate directly to Adobe Connect or use external authentication. Depending on your environment and server configuration, you might also use combinations of these options.

**Cookie management** When a user logs in, Adobe Connect returns a cookie that identifies the user's session. You need to pass the cookie back to the server on all calls made to the server during the user's session. Then, when the user logs out, the server makes the cookie expire and you should invalidate it.

In your development environment, you can use a code library that manages cookies for you. The process of logging in and managing a user's session varies according to whether you use a cookie management library or manage the user's session yourself.

**Licensed server or hosted account** Your organization might have a licensed Adobe Connect server within your firewall, or you may have an Adobe Connect hosted account at Adobe. Either way, you send XML requests over HTTP or HTTPS, but security requirements and the login process vary. If you are a hosted customer, you can use certain parameters with the `login` action to avoid sending user IDs and passwords over the Internet.

**Direct or external authentication** Whether you are a hosted or licensed customer, your application might authenticate directly to Adobe Connect, or you might authenticate users on your own network, set an identifier in an HTTP request header, and send it to Adobe Connect. The login process varies according to whether you use direct or external authentication.

## Log in to Adobe Connect server

The standard technique for logging a user in to Adobe Connect server uses the `login` action, passing the user's login ID and password. This technique works with both HTTP `GET` and `POST` requests.

You also need to manage the `BREEZESSESSION` cookie the server returns for each user session. If you use a client-side cookie management library, it is much easier to allow it to manage cookies for you than to manage the cookies yourself. If you do not have such a library, call `login` with the `session` parameter, as it is easier and more reliable than setting HTTP header values.

*Note: If you send user passwords to Adobe Connect server, use SSL so passwords are encrypted in transit, even if you have a licensed Adobe Connect server within your own firewall.*

### Log in with cookie management

- 1 Call the `login` action, passing it the user's login ID and password, but no `session` parameter:

**Login and requests**

```
http://example.com/api/xml?action=login&login=bobs@acme.com
&password=football
```

- 2 Parse the response for a status code of `ok`.

If the login is successful, the server returns the `BREEZESSESSION` cookie in the response header:

```
Set-Cookie: BREEZESSESSION=breezbryf9ur23mbokzs8; domain=.macromedia.com; path=/
```

- 3 Allow your cookie management library to manage the `BREEZESSESSION` cookie.

Your client-side library passes the cookie back to the server in a request header on subsequent calls for the remainder of the user's session. You do not need to set the cookie in the request header explicitly. When the user logs out, the cookie expires.

**Log in using the session parameter**

- 1 Before you log the user in, call `common-info` to get the value of the `BREEZESSESSION` cookie:

```
http://example.com/api/xml?action=common-info
```

- 2 Extract the cookie value from the response:

```
<cookie>breezxq66rt43poi3if8</cookie>
```

- 3 Log the user in, specifying the cookie value:

```
http://example.com/api/xml?action=login&login=bobs@acme.com
&password=football&session=breezxq66rt43poi3if8
```

- 4 Parse the response for a status code of `ok`.

- 5 Use the `session` parameter with the same cookie value on subsequent calls for the user, until the user's session ends:

```
https://example.com/api/xml?action=principal-list
&session=breezxq66rt43poi3if8
```

- 6 When the user logs out or the user's session ends, do not reuse the cookie value.

**Log in to a Adobe Connect hosted account**

If you want to log in directly to an Adobe Connect hosted account or multiple hosted accounts, you still use the `login` action, but you need to specify an account ID or domain name, in addition to the user's login ID and password. You can specify a domain name if you want to avoid sending an account ID over the Internet.

With an Adobe Connect hosted account, you cannot use single sign-on or external authentication. You must pass the user's authentication credentials on the Adobe Connect hosted account, not the credentials for an external network.

*Note: It is important to have SSL enabled on your Adobe Connect hosted account, because you are sending user IDs, passwords, and account information over the Internet to your Adobe Connect account hosted at Adobe.*

**Log in to an Adobe Connect hosted account with an account ID**

- 1 Before you log the user in, call `common-info` with the domain name of your Adobe Connect hosted account in either the request URL or the `domain` parameter:

```
http://acme.adobe.com/api/xml?action=common-info
http://adobe.com/api/xml?action=common-info&domain=acme.adobe.com
```

- 2 Parse the response for the values of `cookie` and `account-id`:

```
<cookie>Sbreezdd2dfr2ua5gscogv</cookie>
...
<account account-id="295153" />
```

**Login and requests**

- 3 Collect the user's login ID and password in your application.
- 4 Call the `login` action, adding the user's credentials and the `account-id` and `session` parameters:
 

```
https://example.com/api/xml?action=login&login=joy@acme.com
    &password=happy&account-id=295153&session=Sbreezsd2dfr2ua5gscogv
```
- 5 Parse the response for a status code of `ok`.
- 6 (Optional) If you prefer, you can call `login` before `common-info`, extract the cookie value from the response header, and manage it yourself or using a cookie management library.

**Log in to an Adobe Connect hosted account with a domain name**

- 1 Before you log the user in, call `common-info` with the domain name of your Adobe Connect hosted account in either the request URL or the `domain` parameter:

```
http://acme.adobe.com/api/xml?action=common-info
http://adobe.com/api/xml?action=common-info&domain=acme.adobe.com
```

- 2 Parse the response for the values of `cookie` and `host`:

```
<cookie>breezxq66rt43poi3if8</cookie>
...
<host>https://acme.adobe.com</host>
```

- 3 Extract the domain name from the value of `host`:

```
acme.adobe.com
```

- 4 In your application, collect the user's login ID and password.

Be sure the login ID is the user's Adobe Connect hosted account login ID, not an external one.

- 5 Call `login`, adding the user's credentials and the `domain` and `session` parameters:

```
https://example.com/api/xml?action=login&login=joe
    &password=smith99&domain=acme.adobe.com&session=breezxq66rt43poi3if8
```

The `domain` is equivalent to the `account-id`, but by using it you can avoid sending an account ID over the Internet, especially if you use a non-encrypted connection.

- 6 Parse the response for a status code of `ok`.
- 7 (Optional) If you prefer, you can call `login` before `common-info`, extract the cookie value from the response header, and manage it yourself or using a cookie management code library.

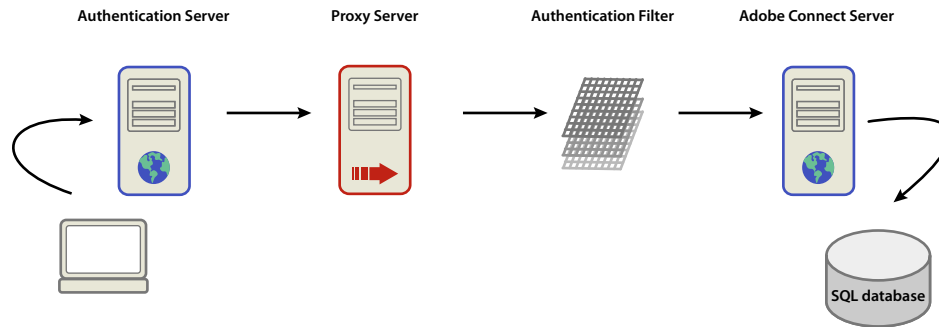
**Log in using HTTP header authentication**

**Note:** The instructions in this section apply only to Adobe Connect server.

Your application can use a trusted central server to authenticate users with single sign-on and pass your network's (here called *external*) authentication to Adobe Connect server, without explicitly passing an Adobe Connect server user ID and password. (For detailed instructions on how to set up and configure HTTP header authentication, see *Adobe Connect Installation and Configuration Guide*).

With HTTP header authentication, a user logs in to your authentication server. Once the user is authenticated, you add an HTTP request header that identifies the user, or configure a proxy server to add the header. The authentication filter on Adobe Connect (named `HeaderAuthenticationFilter`) converts your user identifier to an Adobe Connect login ID and authenticates the user.



**Login and requests**

*Authentication filters convert external authentication credentials to Adobe Connect credentials.*

External authentication works in addition to standard Adobe Connect authentication. Each user who needs to access Adobe Connect server needs a valid Adobe Connect server login and password.

When you send a login request to Adobe Connect server with an external authentication credential:

- The authentication filter intercepts the request and checks for a user on Adobe Connect server with an `ext-login` field that matches your external credential.
- If a match exists, the filter passes your external authentication to Adobe Connect server, and the server logs the user in.
- If no match exists, the filter passes the login request to the server, which displays its login page. The user must then log in to Adobe Connect server.
- If the user logs in successfully, Adobe Connect server updates the `ext-login` field in the user's profile with the external credential from your request. The next time you send a request with the user's external credential, Adobe Connect server finds a match in `ext-login`, and the user does not need to log in to Adobe Connect.
- If the user does not log in successfully, the user is not allowed access to Adobe Connect server applications, content, or meetings.

The steps that follow describe how to call `login` when you use HTTP header authentication.

**Log in to Adobe Connect server using HTTP header authentication**

- 1 Configure your network servers and Adobe Connect server for HTTP header authentication using the instructions in *Adobe Connect Installation and Configuration Guide*.
- 2 In `[your server directory]/appserv/conf/WEB-INF/web.xml`, remove comment tags around the `filter-mapping` element for `HeaderAuthenticationFilter` and add comment tags around any other `filter-mapping` elements:

```
<filter-mapping>
  <filter-name>HeaderAuthenticationFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
<!--
<filter-mapping>
<filter-name>NtlmAuthenticationFilter</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
-->
```

- 3 In the `filter` element for `HeaderAuthenticationFilter`, enable the `/api/` pattern for request URLs. You have two choices for how to do this:

**If your application uses the XML API and any Adobe Connect applications** In the `filter` element for `HeaderAuthenticationFilter`, use comment tags to disable the `init-param` element with a `param-value` of `/api/`:

**Login and requests**

```

<!--
<init-param>
  <param-name>ignore-pattern-0</param-name>
  <param-value>/api/</param-value>
</init-param>
-->

```

**If your application uses only the XML API** Change the `filter-mapping` element for your filter type to use the URL pattern `/api/*` instead of `/*`:

```

<filter-mapping>
  <filter-name>HeaderAuthenticationFilter</filter-name>
  <url-pattern>/api/*</url-pattern>
</filter-mapping>

```

Then, in the `filter` element for your filter type, add comment tags around all `init-param` elements with a `param-name` of `ignore-pattern-x`:

```

<filter>
  <filter-name>HeaderAuthenticationFilter</filter-name>
  <filter-class>
    com.macromedia.airspeed.servlet.filter.HeaderAuthenticationFilter
  </filter-class>
  <!--
  <init-param>
    <param-name>ignore-pattern-0</param-name>
    <param-value>/api/</param-value>
  </init-param>
  ...
  <init-param>
    <param-name>ignore-pattern-4</param-name>
    <param-value>/servlet/testbuilder</param-value>
  </init-param>
  -->
</filter>

```

- 4 Configure Adobe Connect server so that users are created with the field `ext-login` set to the external user ID you send (see *Adobe Connect Installation and Configuration Guide* for details).

By default, `ext-login` has the same value as `login`, the Adobe Connect server login ID.

- 5 Once your system authenticates the user, create a `login` request. Add the parameter `external-auth=use`, but no `login` or `password` parameters:

```
https://example.com/api/xml?action=login&external-auth=use
```

- 6 Add your authenticated user ID to the HTTP request header. By default, use the header name `x-user-id`:

```
x-user-id: joesmith
```

You can specify a different header name by setting a value for `HTTP_AUTH_HEADER` in the `custom.ini` file. You can also configure a proxy server to set the HTTP header value. See *Adobe Connect Installation and Configuration Guide* for details of either.

- 7 Parse the response for a status code of `ok`.
- 8 Handle the `BREEZESSESSION` cookie value returned in the response header. You have two choices for how to do this:
  - If you use a client library that manages cookies** Allow your library to extract the cookie value, store it, and pass it back to the server on subsequent requests for the user.

**If you manage cookies yourself** Extract the value of the `BREEZESSESSION` cookie from the response header. Store it and pass it back to the server in the `session` parameter of all subsequent actions you call for the same user, as long as the user's session is valid:

```
https://example.com/api/xml?action=principal=list&session=breezs7zuepmy9wh2tseu
```

Be sure not to reuse the cookie value when the user's session ends.

## Send a request in an XML document

At times, you may prefer to send an HTTP `POST` request to the server to make sure the data is secure and not visible in transit. In that case, specify the action name and parameters in an XML document.

### Make an XML document request

- 1 Create an XML document with the root element `params` and `param` child elements for the action name and each parameter:

```
<params>
  <param name="action">login</param>
  <param name="login">jon@doe.com</param>
  <param name="password">foobar</param>
</params>
```

- You can only send one action in the `params` root element. You cannot batch multiple actions to be executed sequentially.
  - The XML document you send must be valid and well-formed. Try validating the document in an XML editor before you send it.
- 2 Write code that sends an HTTP `POST` request to Adobe Connect and receives an XML response.  
The specific code will vary according to your programming language and development environment.
  - 3 In your code, send the XML document to Adobe Connect in the body of the HTTP `POST` request.
    - Read the XML document into the request.
    - Be sure to set a `content-type` header of `text/xml` or `application/xml`.

## Parse a response with XPath

When you receive an XML response from Adobe Connect, you need to be able to parse it to extract the XML elements you need.

If you are working in a language such as Java™, with an XML parser (such as Xerces or JDOM) installed, you can parse through an XML response, select values from nodes, and then use those values.

### Use XPath to parse a response

- ❖ Write a method that calls one or more actions. Create an instance of the XPath class so that you can use the XPath expressions. Call the actions, read the XML response, and use XPath syntax to select the values you need:

```
public String scoUrl(String scoId) throws XMLApiException {
    try {
        Element e = request("sco-info", "sco-id=" + scoId);
        if(! (codePath.valueOf(e).equalsIgnoreCase("ok")))
            return "";
        XPath xpath = XPath.newInstance("//url-path/text()");
        String path = ((Text) xpath.selectSingleNode(e)).getText();

        e = request("sco-shortcuts", null);
        xpath = XPath.newInstance("//domain-name/text()");
        String url = ((Text) xpath.selectSingleNode(e)).getText();

        return url + "/" + path.substring(1) + "?session=" + breezesession;
    } catch (JDOMException jde) {
        throw new XMLApiException(PARSE_ERROR, jde);
    }
}
```

You can also use string pattern matching to check for a status code of `ok`. A successful action always returns this response:

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
</results>
```

You can check the response for the pattern `ok` or `code="ok"` .

## Parse an error response

When an API action completes successfully, it returns a status code of `ok`. If the call is not successful, it can also return any of the following status codes:

**invalid** Indicates that the call is invalid in some way, usually invalid syntax.

**no-access** Shows that the current user does not have permission to call the action, and includes a `subcode` attribute with more information.

**no-data** Indicates that there is no data available for the action to return, when the action would ordinarily return data.

**too-much-data** Means that the action should have returned a single result but is actually returning multiple results.

When the status code is `invalid`, the response also has an `invalid` element that shows which request parameter is incorrect or missing:

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="invalid">
    <invalid field="has-children" type="long" subcode="missing" />
  </status>
</results>
```

When the status code is `no-access`, the `subcode` explains why:

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="no-access" subcode="denied" />
</results>
```

All valid values for `code`, `subcode`, and `invalid` are described in [status](#), in the API reference. Your application needs to read and handle status codes and subcodes.

### Handle status codes

- 1 Write a method that parses an XML API response for the status `code` and `subcode`. This is an example in Java:

```
private String getStatus(Element el) throws JDOMException {
    String code = codePath.valueOf(el);
    String subcode = subcodePath.valueOf(el);
    StringBuffer status = new StringBuffer();
    if(null != code && code.length() > 0)
        status.append(code);
    if(null != subcode && subcode.length() > 0)
        status.append(" - " + subcode);
    return status.toString();
}
```

- 2 When you call an action, parse the response for the status.
- 3 If the status is not `ok`, return a `null` value, display the error status code for debugging, or throw an application exception.

The action to take depends on which call you are making and how your application is designed.

## Log a user out

When a user logs out, the user's session ends, and Adobe Connect invalidates the `BREEZESSESSION` cookie by setting it to `null` and using an expiration date that has passed. For example, if you call `logout` on August 29, 2006, you see this `Set-Cookie` method in the response header, setting an empty cookie value and an expiration date a year earlier:

```
Set-Cookie: BREEZESSESSION=;domain=.macromedia.com;expires=Mon, 29-Aug-2005
22:26:15 GMT;path=/
```

If you are managing the `BREEZESSESSION` cookie, invalidate the value so it is not reused after a user logs out.

### Log a user out and invalidate the session cookie

- 1 Call `logout` to log the user out:

```
https://example.com/api/xml?action=logout
```

- 2 Parse for a status code of `ok` to make sure the logout was successful.
- 3 Set the cookie value to `null` or otherwise invalidate it. For example, in this Java code snippet, the `breezesession` variable stores the cookie value and is set to `null`:

```
public void logout() throws XMLApiException {
    request("logout", null);
    this.breezesession = null;
}
```

# Chapter 4: Basics

To get started with Adobe Connect Web Services, you need to understand three key concepts:

- Principals, who are users and groups
- SCOs, which are Shareable Content Objects and represent meetings, courses, and just about any content that can be created on Adobe Connect. SCOs (pronounced *sco*, which rhymes with *snow*) are compatible with the industry standard Shareable Content Object Reference Model (SCORM) specification and can be used with a Learning Management System (LMS).
- Permissions, which define how principals can act on objects

This chapter describes basic tasks you can do with Web Services, regardless of which Adobe Connect applications you have licensed. Many tasks are described as if you are running them in a browser. If you want to make the call from an application, translate the XML request to the language you are working in (for an example of how to do this in Java™, see “[Send XML requests](#)” on page 251).

## Find a principal-id

A principal is a user or group that has a defined permission to interact with a SCO on the server. You can create users and groups for your organization and modify their permissions.

Adobe Connect also has *built-in groups*: Administrators, Limited Administrators, Authors, Training Managers, Event Managers, Learners, Meeting Hosts, and Seminar Hosts. You can add users and groups to built-in groups, but you can’t modify the permissions of built-in groups.

*Note: The built-in groups that are available depend on your account.*

Each Adobe Connect user and group has a `principal-id`. In some API calls, the `principal-id` is called a `group-id` or `user-id` to distinguish it from other values. The value of the ID that identifies a user or group is always the same, regardless of its name. You can check the syntax of any action in “[Action reference](#)” on page 58

### Get the principal-id of a user or group

- 1 Call `principal-list` with a filter:

```
https://example.com/api/xml?action=principal-list&filter-name=jazz doe
```

It is best to use `filter-name`, `filter-login`, or `filter-email` for an exact match. Be careful with `filter-like-name`, as it may affect server performance.

- 2 Parse the `principal` elements in the response for the `principal-id`:

```
<principal principal-id="2006282569" account-id="624520" type="user"
  has-children="false" is-primary="false" is-hidden="false">
  <name>jazz doe</name>
  <login>jazzdoe@example.com</login>
  <email>jazzdoe@newcompany.com</email>
</principal>
```

### Get the principal-id of the current user

- 1 Call `common-info` after the user is logged in:

```
https://example.com/api/xml?action=common-info
```

- 2 Parse the user elements in the response for the user-id:

```
<user user-id="2007124930" type="user">
  <name>jazz doe</name>
  <login>jazz@doe.com</login>
</user>
```

Here, the principal-id is called user-id, because it always represents a user who is authenticated to Adobe Connect. A group cannot log in to the server. You can pass the user-id value as a principal-id in other actions.

## List principals or guests

A principal with a type of user is a registered Adobe Connect user, while a user with a type of guest has entered a meeting room as a guest. The server captures information about the guest and gives the guest a principal-id.

### List all principals on the server

- 1 Call `principal-list` with no parameters:

```
https://example.com/api/xml?action=principal-list
```

This call returns all Adobe Connect users, so be prepared for a large response.

- 2 Parse the principal elements in the response for the values you want:

```
<principal principal-id="2006282569" account-id="624520" type="user"
  has-children="false" is-primary="false" is-hidden="false">
  <name>jazz doe</name>
  <login>jazzdoe@example.com</login>
  <email>jazzdoe@newcompany.com</email>
</principal>
```

### List all guests on the server

- 1 Call `report-bulk-users`, filtering for a type of guest:

```
https://example.com/api/xml?action=report-bulk-users&filter-type=guest
```

- 2 Parse the row elements in the response:

```
<row principal-id="51157227">
  <login>joy@acme.com</login>
  <name>joy@acme.com</name>
  <email>joy@acme.com</email>
  <type>guest</type>
</row>
```

### List all users who report to a specific manager

When you call `principal-info` with a principal-id, the response shows the principal. If the principal is a user who has a manager assigned in Adobe Connect, the response also shows data about the principal's manager in a `manager` element:

```
<manager account-id="624520" disabled="" has-children="false" is-hidden="false" is-
primary="false" principal-id="2006282569" type="user">
  <ext-login>jazzdoe@example.com</ext-login>
  <login>jazzdoe@example.com</login>
  <name>jazz doe</name>
  <email>joy@example.com</email>
  <first-name>jazz</first-name>
  <last-name>doe</last-name>
  <x-2006293620>23456</x-2006293620>
  <x-2007017651>chicago</x-2007017651>
</manager>
```

You can use the manager's principal-id with principal-list to list all users who are assigned to the manager.

- 1 Call principal-list, filtering on manager-id:

```
https://example.com/api/xml?action=principal-list&filter-manager-id=2006282569
```

- 2 Parse the response for the principal elements:

```
<principal principal-id="2006258745" account-id="624520" type="user" has-children="false"
is-primary="false" is-hidden="false" manager-id="2006282569">
  <name>Pat Lee</name>
  <login>plee@mycompany.com</login>
  <email>plee@mycompany.com</email>
</principal>
```

## Create users

To create a new user, you need Administrator privilege. Adobe recommends that you create a user who belongs to the admins group for your application to use to make API calls that require Administrator privilege.

### Create a new user and send a welcome e-mail

- 1 In your application, log in as an Administrator user.

See [Log in from an application](#) for various ways to log in.

- 2 Call <<UNRESOLVED XREF>> principal-update with at least these parameters:

```
https://example.com/api/xml?action=principal-update
  &first-name=jazz&last-name=doe&login=jazz99@doe.com&password=hello
  &type=user&send-email=true&has-children=0&email=jazz99@doe.com
```

The type must be user, has-children must be 0 or false, send-email must be true, and email must have a valid e-mail address.

The server sends a welcome e-mail with login information to the user's e-mail address.

- 3 Parse the principal element in the response for the user's principal-id:

```
<principal type="user" principal-id="2007184341" has-children="0"
account-id="624520">
  <login>jammdoe@example.com</login>
  <ext-login>jammdoe@example.com</ext-login>
  <name>jamm doe</name>
</principal>
```



**Create a new user without using an e-mail address as a login ID**

- 1 In Connect Central, navigate to Administration > Users and Groups > Edit Login and Password Policies. Make sure that Use E-mail Address as the Login is set to No.
- 2 In your application, log in as an Administrator user.
- 3 Call <<UNRESOLVED XREF>> `principal-update` to create the new user, passing both `login` and `email` parameters:

```
https://example.com/api/xml?action=principal-update&first-name=jazz
&last-name=doe&login=jazz&email=jazzdoe@company.com
&password=nothing&type=user&has-children=0
```

- 4 Parse the response for the `principal-id` of the new user:

```
<principal type="user" principal-id="2007184341" has-children="0"
  account-id="624520">
  <login>jazzdoe@example.com</login>
  <ext-login>jazzdoe@example.com</ext-login>
  <name>jazz doe</name>
</principal>
```

In the response, `ext-login` has the same value as `login` by default, until the user logs in successfully using external authentication (see [Log in using HTTP header authentication](#)).

## Update users

Once you create users, you often need to update their information. You can update standard fields that Adobe Connect defines for users by calling `principal-update` with the user's `principal-id`. The standard fields include `email`, `login`, `first-name`, and `last-name`.

If you have defined custom fields for the principal, use `acl-field-update` to update them.

You need Administrator privilege to update users, so your application must first log in as a user in the `admins` group. You cannot log in as the user and then have the user update his or her own profile.

**Update standard user information**

- 1 Log in as an Administrator user.
- 2 Call [principal-list](#) with a filter to get the user's `principal-id` (see [Find a principal-id](#)).
- 3 Call <<UNRESOLVED XREF>> `principal-update` to update the user:

```
https://example.com/api/xml?action=principal-update
&principal-id=2006282569&email=jazzdoe@newcompany.com
```

- 4 Parse the response for a status code of `ok`.

**Update custom field values for a user**

- 1 Log in as an Administrator user.
- 2 Call `custom-fields` to get the `field-id` of the custom field:
 

```
https://example.com/api/xml?action=custom-fields
```
- 3 Get the `principal-id`, `sco-id`, or `account-id` you want to update.

This value is the `acl-id` you pass to `acl-field-update`.

- 4 Call `acl-field-update` to update the value of the custom field:

```
https://example.com/api/xml?action=acl-field-update&field-id=x-2007396975&acl-id=2006258745&value=44444
```

## Create custom fields

Custom fields are additional data fields that you define. You can define up to eight custom fields on a principal or SCO using `custom-field-update`.

Once you define the custom field, by default you can set its value either by editing the value in Adobe Connect Central or by calling `custom-field-update`.

To specify that the value can only be updated through the API, call `custom-field-update` with the parameter `object-type=object-type-read-only`.

### Define a custom field and set it on a user

- 1 First, create the field with `custom-field-update`:

```
https://example.com/api/xml?action=custom-field-update
&object-type=object-type-principal&permission-id=manage
&account-id=624520&name=Location&comments=adobe%20location
&field-type=text&is-required=true&is-primary=false&display-seq=9
```

The `name` field defines the field name as your application displays it, so use appropriate spelling and capitalization. The custom field in this example is defined for all Adobe Connect principals.

- 2 Parse the `field` element in the response for the `field-id`:

```
<field field-id="2007184366" object-type="object-type-principal"
display-seq="9" account-id="624520" is-primary="false"
permission-id="manage" is-required="true" field-type="text">
<comments>test</comments>
<name>Country</name>
</field>
```

- 3 Get the `principal-id` of the user (see [Find a principal-id](#)).

- 4 Call `acl-field-update` to set the value of the field, passing a `field-id`, the user's `principal-id` as `acl-id`, and a value:

```
https://example.com/api/xml?action=acl-field-update
&acl-id=2006258745&field-id=2007017474&value=San%20Francisco
```

- 5 Parse the response for a status code of `ok`.

## Create groups

To add users to groups, you need to call `principal-update` as your application's Administrator user.

### Add a user to a group

- 1 Log in as your application's Administrator user.
- 2 (Optional) If the user does not yet exist, create the user with `<<UNRESOLVED XREF>>` `principal-update`:

**Basics**

```
https://example.com/api/xml?action=principal-update
&first-name=jazzwayjazz&last-name=doe&login=jazz@doe.com
&password=nothing&type=user&has-children=0
```

3 (Optional) Parse the response for the new user's principal-id.

4 If the user already exists, call [principal-list](#) to get the user's principal-id:

```
https://example.com/api/xml?action=principal-list&filter-type=user
```

5 Parse the response for the principal-id:

```
<principal principal-id="5611980" account-id="624520" type="user"
  has-children="false" is-primary="false" is-hidden="false">
  <name>Joy Black</name>
  <login>joy@acme.com</login>
  <email>joy@acme.com</email>
</principal>
```

6 Call `principal-list` again to get the group's principal-id:

```
https://example.com/api/xml?action=principal-list&filter-type=group
```

7 Call `group-membership-update` with `is-member=true` to add the user to the group:

```
https://example.com/api/xml?action=group-membership-update
&group-id=4930296&principal-id=2006258745&is-member=true
```

- The `principal-id` is the user's principal-id.
- The `group-id` is the group's principal-id.
- The parameter `is-member` must be true.

**Check whether a specific user is in a group**

1 Call `principal-list` with a `group-id`, `filter-is-member`, and a filter that identifies the principal:

```
https://example.com/api/xml?action=principal-list&group-id=624523
&filter-is-member=true&filter-like-name=bob
```

2 Parse for a principal element in the response. A successful response looks like this:

```
<principal-list>
  <principal principal-id="624660" account-id="624520" type="user"
    has-children="false" is-primary="false" is-hidden="false">
    <name>Bill Jones</name>
    <login>bjones@acme.com</login>
    <email>bjones@acme.com</email>
    <is-member>true</is-member>
  </principal>
</principal-list>
```

If the user is not a group member, the `principal-list` element is empty:

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <principal-list />
</results>
```

**Check which users are in a group**

1 To get the group's principal-id, call [principal-list](#) with filters:

**Basics**

```
https://example.com/api/xml?action=principal-list&filter-type=group
&filter-name=developers
```

With `filter-type` and `filter-name`, `principal-list` should return a unique match.

**2** Parse the response for the `principal-id`:

```
<principal principal-id="2007105030" account-id="624520"
  type="group" has-children="true" is-primary="false"
  is-hidden="false">
  <name>developers</name>
  <login>developers</login>
  <is-member>false</is-member>
</principal>
```

**3** Call `principal-list` again, with the `principal-id` as a `group-id` and `filter-is-member=true`:

```
https://example.com/api/xml?action=principal-list&group-id=2007105030
&filter-is-member=true
```

**4** Parse the response for the principal elements:

```
<principal principal-id="5698354" account-id="624520" type="group"
  has-children="true" is-primary="false" is-hidden="false">
  <name>Bob Jones</name>
  <login>bobjones@acme.com</login>
  <is-member>true</is-member>
</principal>
```

**List all groups a user belongs to****1** Call `principal-list` with the user's `principal-id` and `filter-is-member=true`:

```
https://example.com/api/xml?action=principal-list
&principal-id=2006258745&filter-is-member=true
```

**2** Parse the response for the principal elements:

```
<principal principal-id="5698354" account-id="624520" type="group"
  has-children="true" is-primary="false" is-hidden="false">
  <name>Bob Jones</name>
  <login>bobjones@acme.com</login>
  <is-member>true</is-member>
</principal>
```

## Find SCOs

All objects on Adobe Connect are Shareable Content Objects, or SCOs. The word *Shareable* comes from learning management systems in which content is combined into courses or curriculums and shared among them.

On the server, a SCO can be any content object that is combined with other content objects into a course or curriculum. Courses, curriculums, presentations, and other types of content are SCOs. Meetings, events, folders, trees, links, graphics files, or any other object are also SCOs.

Each SCO has a unique integer identifier called a `sco-id`. The `sco-id` is unique across the entire server. On a Adobe Connect hosted account, the `sco-id` is unique across all accounts.

Each SCO also has a `type`, such as `content`, `course`, `meeting`, and so on. You can see the `sco-id` and `type` values in the response from `sco-info` or other actions:

**Basics**

```
<sco account-id="624520" disabled="" display-seq="0" folder-id="2006258747"
  icon="producer" lang="en" max-retries="" sco-id="2006334909"
  source-sco-id="" type="content" version="1">
```

**Characteristics of SCOs**

When you study the XML responses of various calls, you notice more characteristics of SCOs:

- A SCO's identifier is called a `sco-id` in some actions, but can also be called `folder-id`, `acl-id`, or another name in other actions. It's the same unique ID.
- Each SCO can be accessed by various principals, either users or groups. The specific principals who can access a SCO are defined in access control lists, or ACLs.
- Each SCO has a unique URL, with two parts: a domain name (like `http://example.com`) and an URL path (like `/f2006123456/`). You can concatenate these to form the full URL that accesses the SCO.
- Each SCO has a navigation path that describes where it resides in the folder hierarchy.
- Each SCO has a permission defined for each principal who can access it.
- Some SCOs have description fields, which are text strings that give you information about the SCO.

Often you need to find the ID of a SCO or some information about it. SCOs are arranged in a specific folder hierarchy where folders have names that indicate whether they are at the top level, contain shared content or templates, or hold user content and templates.

When you call `sco-shortcuts`, it returns a list of folders. Notice that folders have different types:

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <shortcuts>
    <sco tree-id="624530" sco-id="2006258751" type="my-meeting-templates">
      <domain-name>http://example.com</domain-name>
    </sco>
    <sco tree-id="624530" sco-id="2006258750" type="my-meetings">
      <domain-name>http://example.com</domain-name>
    </sco>
    <sco tree-id="624529" sco-id="624529" type="meetings">
      <domain-name>http://example.com</domain-name>
    </sco>
    <sco tree-id="624530" sco-id="624530" type="user-meetings">
      <domain-name>http://example.com</domain-name>
    </sco>
    ...
  </shortcuts>
</results>
```

The folders shown in this example happen to be for meetings, but folders for other types of SCOs follow a similar pattern. Each folder type stores certain types of objects, with certain access privileges, as follows:

**content, courses, meetings, events, seminars** These are shared folders, such as Shared Meetings, Shared Training, and so on. The Adobe Connect Administrator has access to this folder. The Administrator can assign Manage permission to any user, but only members of the built-in group associated with the folder can create new content or meetings within it.

**user-content, user-meetings, user-courses, user-events** These folders each contain a folder for each user who can create content within it (for example, one folder for each meeting host or training developer).

**my-courses, my-events, my-meetings, my-meeting-templates, my-content** Users create their own content in these folders and have Manage permission on the content. For example, meeting hosts create meetings in their `my-meetings` folder and have Manage permission on those meetings.

**shared-meeting-templates** This folder is within the Shared Meetings folder, contains meeting templates, and inherits permissions from Shared Meetings.

You can list the contents of any folder to get information about a specific SCO. When you need to search for a SCO but do not have a `sco-id`, move through folders using `sco-shortcuts` and `sco-expanded-contents`. Do not use `sco-search`, as it returns only certain types of SCOs.

### Find a SCO when you do not know the sco-id

- 1 Call `<<UNRESOLVED XREF>> sco-shortcuts` to get a list of root folders on Adobe Connect:

```
https://example.com/api/xml?action=sco-shortcuts
```

- 2 Parse the response for a `type` of the root folder that would logically contain the SCO, for example, `my-courses` for a course the user has created.

- 3 Parse the resulting `sco` element for a `sco-id`:

```
<sco tree-id="4930295" sco-id="2006258748" type="my-courses">
  <domain-name>http://example.com</domain-name>
</sco>
```

- 4 Create a call to `<<UNRESOLVED XREF>> sco-expanded-contents` to list the contents of the folder, adding an exact match filter, if possible:

```
https://example.com/api/xml?action=sco-expanded-contents
&sco-id=2006258748&filter-name=All About Web Communities
```

You have several choices of filters:

- An exact match filter on name or `url-path` (like `filter-name` or `filter-url-path`), if you know the name or URL of the SCO.
- A greater-than or less-than date filter (`filter-gt-date` or `filter-lt-date`) on `date-begin`, `date-created`, or `date-modified`, if you know one of those dates.
- A partial name filter (like `filter-like-name`), if you do not know the exact SCO name. However, using this filter might affect system performance.

- 5 Parse the response for the `sco-id`:

```
<sco depth="1" sco-id="2006745671" folder-id="2006258748" type="folder"
  icon="folder" lang="en" source-sco-id="2006745669" display-seq="0"
  source-sco-type="14">
  <name>A Day in the Life Resources</name>
  <url-path>/f28435879/</url-path>
  <date-created>2006-06-12T14:47:59.903-07:00</date-created>
  <date-modified>2006-06-12T14:47:59.903-07:00</date-modified>
</sco>
```

### Get information about a SCO

- 1 Call `sco-info` with the `sco-id`:

```
https://example.com/api/xml?action=sco-info&sco-id=2006745669
```

- 2 Parse the response for name, `url-path`, or any other value:

**Basics**

```
<sco account-id="624520" disabled="" display-seq="0"
  folder-id="2006258748" icon="curriculum" lang="en" max-retries=""
  sco-id="2006745669" source-sco-id="" type="curriculum" version="0">
  <date-begin>2006-06-12T14:45:00.000-07:00</date-begin>
  <date-created>2006-06-12T14:47:59.903-07:00</date-created>
  <date-modified>2006-06-12T14:47:59.903-07:00</date-modified>
  <name>A Day in the Life</name>
  <url-path>/day/</url-path>
</sco>
```

**Construct the URL to a SCO****1 Call sco-shortcuts:**

```
https://example.com/api/xml?action=sco-shortcuts
```

**2 Parse the response for the domain-name value in any sco element:**

```
<sco tree-id="624530" sco-id="2006258750" type="my-meetings">
  <domain-name>http://example.com</domain-name>
</sco>
```

**3 Call sco-info with the sco-id:**

```
https://example.com/api/xml?action=sco-info&sco-id=2006334909
```

**4 Parse the response for the url-path:**

```
<sco account-id="624520" disabled="" display-seq="0"
  folder-id="2006258747" icon="producer" lang="en"
  max-retries="" sco-id="2006334909" source-sco-id=""
  type="content" version="1">
  <date-created>2006-05-11T12:00:02.000-07:00</date-created>
  <date-modified>2006-05-16T15:22:25.703-07:00</date-modified>
  <name>Test Quiz</name>
  <url-path>/quiz/</url-path>
  <passing-score>10</passing-score>
  <duration>15100.0</duration>
  <section-count>6</section-count>
</sco>
```

The `url-path` has both leading and trailing slashes. You can take the `url-path` from `report-my-meetings`, `report-my-training`, or any call that returns it.

**5 Concatenate the url-path with the domain-name:**

```
http://example.com/f2006258748/
```

## Download files

You can download zip files from Adobe Connect to a user's local computer. A zip file is a SCO. To download it, you need to construct a download URL to the zip file, which looks like this:

```
http://server-domain/url-path/output/url-path.zip?download=zip
```

You probably already know the domain name of your server (such as `example.com`). If you do not, you can get it by calling `sco-shortcuts`.

### Download a zip file from the server

- 1 Call `sco-shortcuts`:

```
https://example.com/api/xml?action=sco-shortcuts
```

- 2 Extract any domain-name value from the response:

```
http://example.com
```

- 3 Call `sco-info` with the `sco-id` of the zip file:

```
https://example.com/api/xml?action=sco-info&sco-id=2006258747
```

The SCO is the entire zip file.

- 4 Parse the response for the `url-path` element:

```
<sco account-id="624520" disabled="" display-seq="0" folder-id="624522"
      icon="folder" lang="en" max-retries="" sco-id="2006258747"
      source-sco-id="" type="folder" version="1">
  <date-created>2006-04-18T10:21:47.020-07:00</date-created>
  <date-modified>2006-04-18T10:21:47.020-07:00</date-modified>
  <name>joy@acme.com</name>
  <url-path>/f124567890</url-path>
</sco>
```

- 5 Construct the download URL, for example:

```
https://example.com/quiz/output/quiz.zip?download=zip
```

Be sure to remove the trailing slash from the `url-path` value before adding `.zip` to it (so you have a value like `/quiz.zip`, not `/quiz/.zip`).

## Check permissions

Permissions define the ways in which a principal can interact with a SCO.

A permission mapping, indicating what permissions a principal has for a particular SCO, is called an *access control list* or ACL. An ACL consists of three pieces of information:

- The ID of a principal (a `principal-id`).
- The ID of a SCO, account, or principal being acted on. In permission calls, it's called an `acl-id`. In other calls, the ID might be called a `sco-id`, `account-id`, or `principal-id`.
- A keyword that indicates the permission level the principal has, which is one of the valid values in `permission-id`.

### Check the permission a principal has on a SCO

- 1 Call `permissions-info` with both an `acl-id` and `principal-id`:

```
https://example.com/api/xml?action=permissions-info&acl-id=2006334909
&principal-id=2006258745
```

To check for permissions on a SCO, the `acl-id` is a `sco-id`. The `acl-id` can also be a `principal-id` or `account-id`.

- 2 Parse the response for a `permission-id`:



```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <permission acl-id="2007035246" permission-id="view"
    principal-id="2006258745" />
</results>
```

If a principal does not have an explicit permission to the SCO (in other words, if `permission-id=""`), the principal's permissions on the SCO's parent object apply.

### Check all principals' permissions on a SCO

- 1 Call `permissions-info` with an `acl-id`, but no `principal-id`:

```
https://example.com/api/xml?action=permissions-info&acl-id=2006293572
```

- 2 Iterate through the `principal` elements and parse them for `permission-id` values:

```
<principal principal-id="2596608" is-primary="false" type="user"
  has-children="false" permission-id="view">
  <name>Jay Arnold</name>
  <login>jay@example.com</login>
</principal>
```

The valid permission values are listed in [permission-id](#).

# Chapter 5: Meetings

Custom applications can display, create, and delete Adobe® Connect™ meetings in a web application, portal, or other environment.

## Using web services with Adobe Connect meetings

Custom applications can display, create, and delete Adobe® Connect™ meetings in a web application, portal, or other environment.

When users click a meeting room URL, they enter Adobe Connect, which hosts the meeting room. Adobe Connect then streams audio, video, and rich media content to the meeting room users.

Adobe recommends the following actions for meeting applications:

**report-my-meetings** To display a user's meetings.

**sco-update** To create a meeting room or update information about it.

**permissions-update** To add a host, presenter, and participants to a meeting.

**report-bulk-consolidated-transactions** To calculate meeting usage, especially the amount of time each user has spent in the meeting.

**report-quiz-interactions** To get the results of a meeting poll.

Some actions that handle meetings require Administrator privilege, as noted in the task instructions. Create a Adobe Connect user who is a member of the `admins` group for your application to use to make these calls.

## Find meetings

You often need to locate the `sco-id` of a meeting so that you can invite users, get report information about it, or update it in some other way.

You should understand the structure of folders in which meetings can be stored. By default, meetings are stored in the host's My Meetings folder (called `my-meetings` in the API). For more details on the folder structure, see

[Characteristics of SCOs](#).

### Find the sco-id of a meeting

- 1 Call `sco-shortcuts`:

```
https://example.com/api/xml?action=sco-shortcuts
```

- 2 Parse the response for the `sco-id` of a meetings folder that is likely to contain the meeting:

```
<sco tree-id="624530" sco-id="624530" type="user-meetings">
  <domain-name>http://example.com</domain-name>
</sco>
```

The folder name should be `meetings`, `user-meetings`, or `my-meetings`. Use a folder as far down the tree as you can.

- 3 Call `sco-contents` on the folder, adding a filter or two to reduce the response:

**Meetings**

```
https://example.com/api/xml?action=sco-contents&sco-id=2006258750
&filter-type=meeting&filter-name=Intro to Film
```

- The more specific you can make the filters, the better. Good filters to use are `filter-name`, `filter-url-path`, or a date filter. Be careful with using `filter-like-name`, as it might affect system performance.
- You can also call `sco-expanded-contents` to list subfolders and their contents. However, `sco-contents` is better for server performance, if you know the `sco-id` of the folder that contains the meeting.

**4 Parse the response for the `sco-id` of the meeting:**

```
<sco sco-id="2006743452" source-sco-id="-1625529" folder-id="2006258750"
  type="meeting" icon="meeting" display-seq="0" is-folder="0">
  <name>Intro to Film</name>
  <url-path>/film/</url-path>
  <date-begin>2006-06-09T14:00:00.000-07:00</date-begin>
  <date-end>2006-06-09T20:00:00.000-07:00</date-end>
  <date-modified>2006-06-09T14:07:13.767-07:00</date-modified>
  <duration>06:00:00.000</duration>
</sco>
```

**List all meetings on the server**

- ❖ Call `report-bulk-objects` with `type=meeting`:

```
https://example.com/api/xml?action=report-bulk-objects&filter-type=meeting
```

The response has a row element for each meeting, showing the meeting URL, name, and dates:

```
<row sco-id="2007372149" type="meeting">
  <url>/monday/</url>
  <name>Monday Staff Meeting</name>
  <date-created>2006-12-18T14:15:00.000-08:00</date-created>
  <date-end>2006-12-19T02:15:00.000-08:00</date-end>
  <date-modified>2006-12-18T17:38:11.660-08:00</date-modified>
</row>
```

## Display meetings

In your application, you might want to lists of Adobe Connect meetings, such as a user's present or future scheduled meetings.

An application workflow might log a user in and display the user's meetings, or it might add the user to a meeting and then display meetings. Displaying the user's meetings means listing the contents of the `my-meetings` folder.

**Display a user's meetings**

- 1 Log the user in (see [Log in from an application](#)).
- 2 Call `report-my-meetings` to list the user's meetings:

```
https://example.com/api/xml?action=report-my-meetings
```

You can add a filter to reduce the response. For example, you can exclude meetings that have ended:

```
https://example.com/api/xml?action=report-my-meetings
&filter-expired=false
```

- 3 Parse the response for values from the meeting elements:

**Meetings**

```
<meeting sco-id="2007063179" type="meeting" icon="meeting" permission-id="host" active-
participants="0">
  <name>September All Hands Meeting</name>
  <domain-name>example.com</domain-name>
  <url-path>/sept15/</url-path>
  <date-begin>2006-09-15T09:00:00.000-07:00</date-begin>
  <date-end>2006-09-15T18:00:00.000-07:00</date-end>
  <expired>false</expired>
  <duration>09:00:00.000</duration>
</meeting>
```

- 4 Create the URL to the meeting room by concatenating `http://` or `https://`, `domain-name`, and `url-path`.

**Add a user to a meeting and display meetings**

- 1 Log in as your application's Administrator user.
- 2 Get the user's `principal-id` (see [Find a principal-id](#)).
- 3 Get the `sco-id` of the meeting (see [Find meetings](#)).
- 4 Call `permissions-update` to add the user to the meeting:

```
https://example.com/api/xml?action=permissions-update
&acl-id=2006258765&principal-id=2006258745&permission-id=view
```

Use a `permission-id` of `view` for a participant, `mini-host` for presenter, or `host` for a meeting host.

- 5 Log out as the Administrator user, and log in as the user you just added to the meeting.
- 6 Display the user's current meetings:

```
https://example.com/api/xml?action=report-my-meetings
&filter-expired=false
```

## Create meeting room URLs

You have several choices of how to construct the URL to a meeting room. The best action to call depends on how your application is logged in and where you are in your application workflow.

By default, the meeting room is created in the host's `my-meetings` folder.

**Create the URL to a meeting room for which the user is host**

- 1 If you are logged in as a user, and you want to create a URL to a meeting in the user's `my-meetings` folder, call `report-my-meetings`:

```
https://example.com/api/xml?action=report-my-meetings
```
- 2 Parse the response for the values of `domain-name` and `url-path`:

**Meetings**

```
<meeting sco-id="2007063179" type="meeting" icon="meeting"
  permission-id="host" active-participants="0">
  <name>September All Hands Meeting</name>
  <domain-name>example.com</domain-name>
  <url-path>/sept15/</url-path>
  <date-begin>2006-09-15T09:00:00.000-07:00</date-begin>
  <date-end>2006-09-15T18:00:00.000-07:00</date-end>
  <expired>>false</expired>
  <duration>09:00:00.000</duration>
</meeting>
```

- 3 Concatenate the two values and add `http://` or `https://` at the beginning:

```
https://example.com/online/
```

If you are using HTTPS and you do not explicitly add `https://`, the URL defaults to `http://`, and the user might not be able to access the meeting room.

**Create the URL to a meeting room for which the user is not host**

- 1 Get the `sco-id` of the meeting (see [Find meetings](#)).

- 2 Call `sco-info` with the `sco-id`:

```
https://example.com/api/xml?action=sco-info&sco-id=2006258750
```

- 3 Parse the response for the `url-path`:

```
<sco account-id="624520" disabled="" display-seq="0" folder-id="624530"
  icon="folder" lang="en" max-retries="" sco-id="2006258750"
  source-sco-id="" type="folder" version="1">
  <date-created>2006-04-18T10:21:47.020-07:00</date-created>
  <date-modified>2006-04-18T10:21:47.020-07:00</date-modified>
  <name>joy@acme.com</name>
  <url-path>/f1234567890/</url-path>
</sco>
```

- 4 (Optional) If you know the domain name of your Adobe Connect Server account, create the URL using `http://` or `https://`, then the `domain-name`, then the `url-path`.

- 5 If you do not know the domain name, call `common-info`:

```
https://example.com/api/xml?action=common-info
```

- 6 Parse the response for the value of the `host` element.

## Create meetings

A user must be an Administrator to create a Adobe Connect meeting, which means the user is a member of the Meeting Hosts group. In the response from `principal-list`, this group has `type=live-admins`.

A meeting can be *public*, *protected*, or *private*, and to create each, you need to set a specific combination of `principal-id` and `permission-id`:

- Public, equivalent to *Anyone who has the URL for the meeting can enter the room*  
`principal-id=public-access&permission-id=view-hidden`
- Protected, equivalent to *Only registered users and accepted guests can enter the room*  
`principal-id=public-access&permission-id=remove`

**Meetings**

If a meeting is protected, registered users invited as meeting participants can enter by clicking the meeting room URL and logging in. Users who are not invited can log in as guests. The meeting host receives a guest's request to enter (known as *knocking*) and can accept or decline.

- Private, which is equivalent to *Only registered users and participants can enter*. The login page does not allow guests to log in.

```
principal-id=public-access&permission-id=denied
```

**Create a public meeting and add host, presenter, and participants**

- 1 Call [principal-list](#) to check that the user creating the Adobe Connect meeting is a member of the `live-admins` group:

```
https://example.com/api/xml?action=principal-list&group-id=624523
&filter-is-member=true&filter-like-name=bob
```

- 2 Call `sco-shortcuts` to obtain the `sco-id` of the user's `my-meetings` folder:

```
https://example.com/api/xml?action=sco-shortcuts
```

- 3 Parse the response for the `sco` element with `type=my-meetings`:

```
<sco tree-id="624530" sco-id="2006258750" type="my-meetings">
  <domain-name>http://example.com</domain-name>
</sco>
```

- 4 Call `sco-update` to create the meeting room:

```
https://example.com/api/xml?action=sco-update
&type=meeting&name=October All Hands Meeting
&folder-id=2006258750&date-begin=2006-10-01T09:00
&date-end=2006-10-01T17:00&url-path=october
```

The `folder-id` is the `sco-id` of the user's `my-meetings` folder.

- 5 Parse the response for the `sco-id` of the new meeting:

```
<sco folder-id="2006258750" lang="en" account-id="624520"
  type="meeting" icon="meeting" sco-id="2007184134" version="0">
  <date-begin>2006-10-01T09:00</date-begin>
  <date-end>2006-10-01T17:00</date-end>
  <url-path>/october</url-path>
  <name>October All Hands Meeting</name>
</sco>
```

You might want to store the `url-path` to the meeting, if you plan to create a URL to the meeting room later.

- 6 Call [permissions-update](#) to make the meeting public. Use the `sco-id` of the meeting as the `acl-id`:

```
https://example.com/api/xml?action=permissions-update&acl-id=2007018414
&principal-id=public-access&permission-id=view-hidden
```

- 7 Call `permissions-update` to add a host, a presenter, and participants:

```
https://example.com/api/xml?action=permissions-update
&principal-id=2006258745&acl-id=2007018414&permission-id=host
```

- Use a `permission-id` of `host` for the meeting host.
- Use `mini-host` for the presenter.
- Use `view` for meeting participants.

**Meetings**

- You can specify multiple trios of `principal-id`, `acl-id`, and `permission-id` on one call to `permissions-update`.

8 Create the URL to the meeting room (see [Create meeting room URLs](#)).

**Create a private meeting and add host, presenter, and participants**

- 1 Log in as your application's Administrator user.
- 2 Follow the steps for creating a public meeting, but set the meeting permission to private:

```
https://example.com/api/xml?action=permissions-update&acl-id=2007018414
&principal-id=public-access&permission-id=denied
```

- 3 Call `permissions-update` again to add a host, a presenter, and guests.
- 4 Create the URL to the meeting room (see [Create meeting room URLs](#)).

## Set or reset a meeting passcode

By default, when meeting hosts create meetings, they can set a passcode that users must enter to join the meeting. In Connect Central, Account Administrators can enable and disable the ability to enforce passcodes; the ability to enforce passcodes is disabled by default.

Use the Web Services API to do the following:

- Enable and disable the ability of meeting hosts to enforce passcodes for meetings.
- Set, reset, or remove the passcode for a meeting.
- Check whether a meeting has a passcode
- View a list of meetings that require passcodes

**Enable and disable the ability to passcode protect meeting rooms**

To enable the passcode protect option for an account, call the following API:

```
http://<server>/api/xml?action=meeting-feature-update&account-id=<acc_id>&feature-id=fid-
meeting-passcode-notallowed&enable=false
```

To disable the passcode protect option, pass `enable=true`.

**Note:** Only administrators can call `meeting-feature-update` to enable or disable a meeting feature.

**Check whether the enforce passcode option is enabled for an account**

Call the `meeting-feature-info` API:

```
http://<server>/api/xml?action=meeting-feature-info&account-id=<acc_id>
```

If the result list contains `feature-id=fid-meeting-passcode-notallowed`, the passcode option is enabled. Otherwise, the passcode option is not enabled. By default, the passcode options is disabled.

**Set, reset, or remove a passcode**

Call the `acl-field-update` API and pass the `meeting-passcode` parameter:

```
http://<server>/api/xml?action=acl-field-update&acl-id=<sco-id>&field-id=meeting-
passcode&value=<passcode>
```

To remove a passcode, set the `value` parameter to empty:

**Meetings**

```
http://<server>/api/xml?action=acl-field-update&acl-id=<sco-id>&field-id=meeting-passcode&value=
```

**Note:** Only administrators and meeting hosts can call *acl-field-update* with *field-id=meeting-passcode*.

**Check whether a meeting has a passcode**

```
http://<server>/api/xml?action=acl-field-info&filter-field-id=meeting-passcode&acl-id=<sco-id>
```

If a meeting has a passcode, the result is as follows:

```
<results>
<status code="ok"/>
<acl-fields>
<field acl-id="22701" field-id="meeting-passcode"><value>connect12</value></field>
</acl-fields>
</results>
```

If passcode is not set then the result is:

```
<results>
<status code="ok"/>
<acl-fields/>
</results>
```

**Note:** Only administrators can call *acl-field-info*.

**View a list of meetings that require passcodes:**

```
http://<server>/api/xml?action=acl-field-list&field-id=meeting-passcode
```

The result is the list of *acl-ids* (meeting-ids):

```
<results>
<status code="ok"/>
<acl-field-list>
<acl acl-id="21907">
<value>breeze</value>
</acl>
<acl acl-id="22701">
<value>connect12</value>
</acl>
<acl acl-id="21401">
<value>raj</value>
</acl>
</acl-field-list>
</results>
```

**Note:** Only administrators can call *acl-field-list*.

## Create customized meetings

When you create a Adobe Connect meeting, you can assign it a meeting room template that creates a custom layout for the meeting room. If you don't assign a template, the meeting room is created with the default meeting template.

To edit a meeting room template, launch Connect Central and click the template's URL. You can edit the template while it is in a meeting templates folder (either My Templates or Shared Templates), if you have edit privileges on the folder.



**Meetings****Create a meeting room using a template**

1 Log in as your application's Administrator user.

2 Call `sco-shortcuts`:

```
https://example.com/api/xml?action=sco-shortcuts
```

3 Parse the response for the `sco-id` of a folder that contains meeting templates:

```
<sco tree-id="624529" sco-id="-625529" type="shared-meeting-templates">
  <domain-name>http://example.com</domain-name>
</sco>
<sco tree-id="624530" sco-id="2006258751" type="my-meeting-templates">
  <domain-name>http://example.com</domain-name>
</sco>
```

4 Call `sco-contents`, passing it the `sco-id` of the meeting templates folder:

```
https://example.com/api/xml?action=sco-contents&sco-id=2006258751
```

5 Parse the response for the `sco-id` of the meeting template you want.

6 Create the meeting using `sco-update`. Pass it the `sco-id` of the meeting template as a `source-sco-id`:

```
https://example.com/api/xml?action=sco-update&type=meeting
&name=August%20All%20Hands%20Meeting&folder-id=2006258750
&date-begin=2006-08-01T09:00&date-end=2006-08-01T17:00
&url-path=august&source-sco-id=2006349744
```

7 Continue to set permissions for the meeting and add participants, host, and presenter (see [Create meetings](#)).

8 Create the URL to the meeting room (see [Create meeting room URLs](#)).

## Invite users to meetings

Once you create a Adobe Connect meeting and add participants and presenters, you may want to send invitations by e-mail. To send a meeting invitation, you need information about the meeting, including the meeting name, the host's name and e-mail address, the meeting room URL, the date and time of the meeting, and the participant's (or presenter's) name and e-mail address.

You can construct an e-mail message using any technique that works with your user interface. Extract specific information about the meeting using the following steps.

**Send an e-mail to meeting participants**

1 Call `sco-info` with the meeting `sco-id`:

```
https://example.com/api/xml?action=sco-info&sco-id=2006334033
```

2 Parse the response for the meeting name, date, or other values:

**Meetings**

```
<sco account-id="624520" disabled="" display-seq="0"
  folder-id="2006258750" icon="meeting" lang="en" max-retries=""
  sco-id="2007063163" source-sco-id="-1625529" type="meeting"
  version="0">
  <date-begin>2006-08-15T09:00:00.000-07:00</date-begin>
  <date-created>2006-07-27T15:30:43.220-07:00</date-created>
  <date-end>2006-08-15T18:00:00.000-07:00</date-end>
  <date-modified>2006-07-27T15:30:43.220-07:00</date-modified>
  <name>August All Hands Meeting</name>
  <url-path>/august/</url-path>
</sco>
```

3 Construct the URL to the meeting room (see [Create meeting room URLs](#)).

4 Call `permissions-info` to get the `principal-id` values of the presenters or participants, filtering on `permission-id`:

```
https://example.com/api/xml?action=permissions-info
  &acl-id=2007018414&filter-permission-id=mini-host
```

- For a list of presenters, use `permission-id=mini-host`.
- For participants, use `permission-id=view`.

5 Parse the response for the `principal-id` values you want:

```
<principal principal-id="2006282569" is-primary="false" type="user"
  has-children="false" permission-id="view">
<name>jazz doe</name>
<login>jazzdoe@example.com</login>
</principal>
```

6 Call `principal-info` with the `principal-id`:

```
https://example.com/api/xml?action=principal-info
  &principal-id=2006282569
```

7 Extract the name and email values from the response:

```
<principal account-id="624520" disabled="" has-children="false"
  is-hidden="false" is-primary="false" principal-id="2006282569"
  type="user">
  <ext-login>jazzdoe@example.com</ext-login>
  <login>jazzdoe@example.com</login>
  <name>jazz doe</name>
  <email>jazzdoe@newcompany.com</email>
  <first-name>jazz</first-name>
  <last-name>doe</last-name>
  <x-2006293620>E3612</x-2006293620>
  <x-2007017651>San Francisco</x-2007017651>
</principal>
```

8 Call `permissions-info` again, filtering on a `permission-id` of host:

```
https://example.com/api/xml?action=permissions-info&acl-id=2007018414
  &filter-permission-id=host
```

9 Parse the response for the `principal-id`:

**Meetings**

```
<principal principal-id="2006282569" is-primary="false" type="user"
  has-children="false" permission-id="host">
  <name>jazz doe</name>
  <login>jazzdoe@example.com</login>
</principal>
```

10 Call `principal-info`, using the `principal-id`:

```
https://example.com/api/xml?action=principal-info
  &principal-id=2006258745
```

11 Parse the `principal` element of the response for the name and login (or name and email):

```
<principal account-id="624520" disabled="" has-children="false"
  is-hidden="false" is-primary="false" principal-id="2006282569"
  type="user">
  <ext-login>jazzdoe@example.com</ext-login>
  <login>jazzdoe@example.com</login>
  <name>jazz doe</name>
  <email>jazzdoe@newcompany.com</email>
  <first-name>jazz</first-name>
  <last-name>doe</last-name>
  <x-2006293620>E3612</x-2006293620>
  <x-2007017651>San Francisco</x-2007017651>
</principal>
```

These are for the sender of the e-mail, who is the meeting host.

## Remove users from meetings

Occasionally a user is invited to a Adobe Connect meeting as participant or presenter but later needs to be removed from the participant list. Removing the user has various results, depending on whether the meeting is public or private:

- **For a public meeting:** The user's permission (participant, presenter, or host) is removed, but the user can still enter the meeting as a guest.
- **For a private meeting:** The user's permission is removed, and the user can enter only as a guest and with approval from the meeting host.

To remove a user's permission to enter, call `permissions-update` with a special permission value, `permission-id=remove`.

If the meeting is in progress and the user has already entered the room, the user is not removed from the meeting. However, when the user's session times out, the user cannot reenter.

### Remove a user's permission to access a meeting

1 (Optional) Call `permissions-info` to check the principal's permission to enter the meeting:

```
https://example.com/api/xml?action=permissions-info&acl-id=2007018414
```

However, you do not need to know the specific permission the principal has before you remove the permission.

- 2 Get the meeting's `sco-id` (see [Find meetings](#)).
- 3 Get the user's `principal-id` (see [Find a principal-id](#)).
- 4 Call `permissions-update`, using the meeting's `sco-id` as the `acl-id` and `permission-id=remove`:

**Meetings**

```
https://example.com/api/xml?action=permissions-update
&acl-id=2007018414&principal-id=2006258745&permission-id=remove
```

## Calculate meeting usage

Once you create users and Adobe Connect meetings, you may need to calculate meeting usage. Meeting usage is often calculated in one of these ways:

- The time each user spends in a specific meeting, in minutes per user
- The number of concurrent meeting participants

The time a user spends in a meeting is measured by a *transaction*, which is the interaction between a principal and a SCO (in this case, between a user and a meeting). The date and time a transaction begins and ends are returned by `report-bulk-consolidated-transactions`.

### Calculate time spent in meetings per user

- 1 Call `report-bulk-consolidated-transactions`, filtering for meetings and another value to identify the meeting, such as a date:

```
https://example.com/api/xml?action=report-bulk-consolidated-transactions
&filter-type=meeting&filter-gt-date-created=2006-07-01
```

- The second filter can be for the date the transaction began or ended, the `principal-id` of the user, the `sco-id` of a specific meeting, or another valid filter that meets your needs.
- This call returns all transactions that meet the filter criteria. Be prepared for a large response.
- The call also returns only users who logged in to the meeting as participants, not users who entered as guests.

- 2 Parse the `row` elements in the response for `date-created` and `date-closed`:

```
<row transaction-id="2007071217" sco-id="2007071193" type="meeting" principal-
id="2007003123" score="0">
  <name>Thursday Meeting</name>
  <url>/thursday/</url>
  <login>jazz@doe.com</login>
  <user-name>jazzwayjazz doe</user-name>
  <status>completed</status>
  <date-created>2006-08-03T12:33:48.547-07:00</date-created>
  <date-closed>2006-08-03T12:34:04.093-07:00</date-closed>
</row>
```

- 3 In your application, calculate the time difference between the two dates.

One way to do this (in Java™) is to write a utility method that converts the [ISO 8601](#) datetime values returned in the response to a `GregorianCalendar` object. Then, convert each `GregorianCalendar` date to milliseconds, calculate the difference between the creation and closing times, and convert the difference to minutes.

- 4 Repeat for all the meeting transactions that meet your criteria, and total the meeting usage times.

## Check meeting quotas

The number of concurrent meeting participants you can have is determined by your Adobe Connect license. To check your quota for the number of concurrent meeting participants, call `report-quotas` and look for the quota named `concurrent-user-per-meeting-quota` in the response:

```
<quota acl-id="624529" quota-id="concurrent-user-per-meeting-quota" used="0"
limit="unlimited" soft-limit="1000000000">
  <date-begin>2004-03-09T09:45:02.297-08:00</date-begin>
  <date-end>2999-12-31T16:00:00.000-08:00</date-end>
</quota>
```

The quota has both a limit and a soft limit. The soft limit is the concurrency limit purchased for the account. It is the same as the limit, unless you purchase a Burst Pack for meetings, which allows additional participants to join past the limit, on an overage basis.

Without a Burst Pack, Adobe Connect enforces the concurrency limit and participants who try to enter after the quota is reached are rejected. If your limit is 20 attendees, attendee 21 receives a notice that the meeting room is full.

All accounts enforce the quotas that are set when the account is created. Accounts do not allow overages, unless you have a Burst Pack. Furthermore, Burst Packs are only for meetings, not for training or seminars.

### Check your meeting concurrency quota and usage

- 1 Call `report-quotas` to check your quota for concurrent meeting users:

```
https://example.com/api/xml?action=report-quotas
```

- 2 Parse the response for the `quota` element with a `quota-id` value of `concurrent-user-per-meeting-quota`.
- 3 Extract the value of `soft-limit`, the limit defined by your Adobe Connect license.
- 4 Call `report-meeting-concurrent-users` to check the peak number of concurrent meeting participants on your server or in your account:

```
https://example.com/api/xml?action=report-meeting-concurrent-users
```

- 5 Parse the response for the `report-meeting-concurrent-users` element. Read the value of the `max-users` attribute and compare it to the value of `soft-limit`:

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-meeting-concurrent-users max-users="18"
    max-participants-freq="3" />
</results>
```

## Get meeting archives

A Adobe Connect meeting can have one or more recorded archives. If the meeting recurs weekly, for example, it might have an archive for each session.

A meeting archive is identified with `type=content` and `icon=archive`. The `icon` value works as a subcategory of `type`, to identify the type of content.

### List archives for a meeting room

- 1 Get the `sco-id` of the meeting (see [Find meetings](#)).

**Meetings**

- 2 Call `sco-expanded-contents` with the `sco-id` and `filter-icon=archive` to list all archives associated with the meeting:

```
https://example.com/api/xml?action=sco-contents&sco-id=2007018414
&filter-icon=archive
```

- 3 Parse the response for the `sco` element and extract the information you want, such as `name`, `date-created`, or `url-path`:

```
<sco sco-id="2598402" source-sco-id="" folder-id="2598379"
  type="content" icon="archive" display-seq="0" is-folder="0">
  <name>EN - Monday Night Football_0</name>
  <url-path>/p71144063</url-path>
  <date-begin>2004-05-17T15:51:54.670-07:00</date-begin>
  <date-end>2004-05-17T15:54:52.920-07:00</date-end>
  <date-modified>2004-05-17T15:55:00.733-07:00</date-modified>
  <duration>00:02:58.250</duration>
</sco>
```

## Get meeting poll results

To access the results of a poll used during a meeting, use `report-quiz-interactions`. This action returns all poll results, but you can use a filter to reduce the response.

Each multiple-choice response in the poll has an integer identifier, with the first response in the displayed list numbered *0*, the second *1*, and so on.

### Get the results of a meeting poll

- 1 Be sure that the meeting host has closed the poll.

The poll results are cached in the meeting until the poll is closed.

- 2 Get the `sco-id` of the meeting (see [Find meetings](#)).
- 3 Call `report-quiz-interactions`, using the meeting's `sco-id`:

```
https://example.com/api/xml?action=report-quiz-interactions
&sco-id=2007071193
```

- 4 (Optional) Add a filter to reduce the response, for example:

- `filter-response=1` to check all users who made a specific response
- `filter-interaction-id=2007027923` to check all responses to a poll (a meeting might have several polls)

- 5 Parse the response for `response`, `name`, or any other values:

```
<row display-seq="1" transcript-id="2007071200"
  interaction-id="2007027923" sco-id="2007071193" score="0">
  <name>jazz doe</name>
  <sco-name>Thursday Meeting</sco-name>
  <date-created>2006-08-03T12:29:09.687-07:00</date-created>
  <description>What is your favorite color?</description>
  <response>4</response>
</row>
```

## Launch meetings with external authentication

Once a user logs in to your network and you authenticate the user to the Adobe Connect Server using an external authentication credential, you may want to allow the user to enter a meeting as participant or guest without having to log in a second time to Adobe Connect.

### Launch a meeting and let the user enter as participant

1 Once the user is authenticated on your network, log the user in to Adobe Connect (see [Log in using HTTP header authentication](#) for details).

2 Get the value of the BREEZESESSION cookie for the user's session, in one of two ways:

- Call `common-info` and retrieve the value of `cookie` from the response:

```
<cookie>breezma6zor9rdfps8h6a</cookie>
```

- Retrieve the value of the BREEZESESSION cookie from the response header:

```
Set-Cookie: BREEZESESSION=breezqw4vtfarqxf9pk2;  
domain=.macromedia.com;path=/
```

3 Create a meeting room URL (see [Create meeting room URLs](#) for details).

4 Append a `session` parameter and the BREEZESESSION cookie value to the meeting room URL:

```
http://example.com/employeeMeeting/?session=breezbityp829r9ozv5rd
```

5 Open the meeting room URL that has `session` appended. One way to do this is with a JavaScript `onClick` command:

```
<a href="http://example.com/employeeMeeting/"  
onClick="javascript:window.open('http://example.com/employeeMeeting/?session=breezbityp829r9ozv5rd','Breeze','toolbar=no,menubar=no,width=800,height=600,resizable=yes'); return false">http://example.com/employeeMeeting/</a>
```

### Launch a meeting and let the user enter as guest

1 Once the user is authenticated on your network, log the user in to Adobe Connect (see [Log in using HTTP header authentication](#) for details).

2 Get the value of the BREEZESESSION cookie for the user's session, in one of two ways:

- Call `common-info` and retrieve the value of `cookie` from the response:

```
<cookie>breezma6zor9rdfps8h6a</cookie>
```

- Retrieve the value of the BREEZESESSION cookie from the response header after calling `login`:

```
Set-Cookie: BREEZESESSION=breezqw4vtfarqxf9pk2;  
domain=.macromedia.com;path=/
```

3 In your application, create a meeting room URL (see [Create meeting room URLs](#)).

4 Append a `guestname` parameter and the user's guest display name to the meeting room URL:

```
http://example.com/employeeMeeting/?guestName=joy
```

5 Open the meeting room URL that has the `guestname` parameter. One way to do this is with a JavaScript `onClick` command:

```
<a href="http://example.com/employeeMeeting/"  
onClick="javascript:window.open('http://example.com/employeeMeeting/?guestName=joy','Breeze','toolbar=no,menubar=no,width=800,height=600,resizable=yes'); return false">http://example.com/employeeMeeting/</a>
```

## Configure compliance settings

Depending on your organization, you might need to configure your system to ensure compliance with governmental regulations and industry standards regarding communication. You can use Adobe Connect to monitor communication data in many ways. For example, you can disable the use of pods, set Adobe Connect to always or never record meetings, generate transcripts of chat sessions, create a notice that recording is taking place, and more. You can also control user access in several ways. For example, you can distinguish between authenticated and non-authenticated users, restrict access to meetings rooms based on roles, and block guest access to rooms. For more information, see *Adobe Adobe Connect User Guide*.

When you change the settings for these features, the changes take effect when a new meeting is started or when the server is refreshed. The typical refresh interval is 10 minutes. The next meeting that starts after the server is refreshed reflects any new settings.

Changing certain settings through the XML API can affect the use of other features. For example, when the attendee list is disabled ([meeting-feature-update](#)), users cannot create breakout rooms. Therefore, to prevent user confusion, disable the breakout rooms feature at the same time.

### Disabling pods

When you disable pods, the layout of a meeting room is affected and may have more empty white space than you want. Administrators can either resize remaining pods to occupy the empty space (the recommended approach), or create new meeting room templates. Otherwise, after a meeting starts, the host can manually resize pods as they see fit.

If a pod with persistent data, such a Chat pod, is disabled and then re-enabled between different sessions of the same meeting, the contents of the old pod are lost.

#### Disable the Chat and Note pods

- 1 Get the account ID for the account under which the meeting exists.
- 2 Log in using the administrative account.
- 3 Call `meeting-feature-update`, passing `fid-meeting-chat` and `fid-meeting-note` as arguments to the `feature-id` parameter, and setting the `enable` attribute for both parameters to `false`.
- 4 Refresh the server or start a new meeting to see the change.

The following code disables the Chat and Note pods:

```
http://localhost/api/xml?action=meeting-feature-update&account-id=7&feature-id=fid-meeting-chat&enable=false&feature-id=fid-meeting-note&enable=false
```

### Managing chat transcripts

To configure Adobe Connect to generate chat transcripts, select Generate chat transcripts for all meetings in Adobe Connect Central or call [meeting-feature-update](#) with the `feature-id` `fid-chat-transcripts`.

To get a chat transcript, you need the `sco-id` of the chat session. Use a combination of XML APIs to get the `sco-id` of a specific transcript. You can then get the transcript from the following Adobe Connect directory:

```
[RootInstall]/content/account-id/transcript-sco-id/output/.
```

#### Get chat transcripts

- 1 Get the `sco-id` of the chat transcripts tree by calling `sco-shortcuts`:

```
[http://example.com/api/xml?action=sco-shortcuts&account-id=7]
```



**Meetings**

- 2 Parse the response for the chat transcripts `tree-id`:

```
<shortcuts>
  <sco tree-id="10026" sco-id="2006258748" type="chat-transcripts">
    <domain-name>http://example.com</domain-name>
  </sco>
  ...
</shortcuts>
</results>
```

- 3 Get the list of chat transcripts for a particular meeting by calling `sco-contents` with the chat transcripts `tree-id` and the filter `source-sco-id`:

```
[http://example.com/api/xml?action=sco-contents&sco-id=10026&filter-source-sco-id=10458]
```

In the example above, 10026 is the `sco-id` of the chat transcripts tree and 10458 is the `sco-id` of the meeting. (You can get the `sco-id` of the meeting from the URL of the meeting information page.)

The list of SCOs that is returned represents the chat transcripts for the meeting.

- 4 Find the chat transcript in the Adobe Connect directory `[RootInstall]/content/account-id/transcript-sco-id/output/`.

## Forcing meetings to be recorded

You can set up Adobe Connect to record all meetings. Adobe recommends that when meetings are recorded, you show a disclaimer to notify users that the meeting is being recorded.

### Force meetings to be recorded

- 1 Disable the setting that lets hosts control recording (`fid-archive`) and enable automatic recording (`fid-archive-force`) by calling `meeting-feature-update`. Pass the two `feature-id` arguments:

```
https://example.com/api/xml?action=meeting-feature-update&account-id=7&feature-id=fid-archive&enable=false&feature-id=fid-archive-force&enable=true
```

- 2 See “[Setting up disclaimer notices](#)” on page 45.
- 3 Refresh the server or start a new meeting to see the change.

## Setting up disclaimer notices

You can set up a disclaimer notice to appear when a user enters a meeting. A disclaimer notice typically displays boilerplate information for your organization. It advises users of the status of the meeting and the terms of use for the meeting. For example, a disclaimer notice could advise users that the meeting is being recorded, and that users cannot join the meeting unless they accept the notice. By default, this option is disabled.

### Set up a disclaimer notice

- 1 Call `meeting-disclaimer-update` and set the text for the disclaimer notice:

```
https://example.com/api/xml?action=meeting-disclaimer-update&account-id=7&disclaimer=Please note that this meeting is being recorded.
```

- 2 Call `meeting-feature-update` to activate the disclaimer:

```
https://example.com/api/xml?action=meeting-feature-update&account-id=7&feature-id=fid-meeting-disclaimer&enable=true
```

- 3 Refresh the server or start a new meeting to see the change.

## Controlling share settings

You can control settings related to the information that a user can share with other users during a meeting. Call `meeting-feature-update` and pass the appropriate feature ID or multiple feature IDs to enable or disable a share setting. For example, to disable screen sharing, call the following code:

```
https://example.com/api/xml?action=meeting-feature-update&account-id=7&feature-id=fid-meeting-desktop-sharing&enable=false
```

The following table lists the feature IDs for share settings. For a full list of feature IDs, see [feature-id](#).

Share setting	Feature ID
Share a computer screen or control of the screen; share a document or white board	fid-meeting-desktop-sharing
Upload a document to the Share pod	fid-meeting-shared-upload
Upload and manage files using the File Share pod	fid-meeting-file-share
Share a white board	fid-meeting-white-board
Display web pages to attendees	fid-meeting-web-links

# Chapter 6: Training

A custom training application or portal can access Adobe® Connect™ Training to display training courses that are available, enroll users or allow them to self-enroll, list all courses and curriculums the user is enrolled in, and generate various reports.

## Using web services with Adobe Connect Training

A custom training application or portal can access Adobe® Connect™ Training to display training courses that are available, enroll users or allow them to self-enroll, list all courses and curriculums the user is enrolled in, and generate various reports. Adobe Connect Training has two types of training modules: courses and curriculums.

A *course* is content (for example, a presentation) that has a set of enrolled learners with usage tracking for each individual. The course can be delivered and administered independently or as part of a curriculum.

A *curriculum* is a group of courses and other learning content that moves students along a learning path. A curriculum contains primarily Adobe Connect Training courses, but may include other items such as content and meetings. As with courses, you can generate reports to track the progress of enrolled learners as they move through the curriculum. This way, you can ensure that enrollees meet the learning objectives.

Courses and content can both be modules within a curriculum, and a content object can be used in any number of courses and curriculums. In Adobe Connect Training, content objects, courses, and curriculums are all SCOs, and each has a unique `sco-id`. Content objects and courses are combinable and reusable, according to the SCORM standard.

As you develop training applications, Adobe recommends that you use the following XML API actions:

**permissions-update** To enroll users in courses and make sure they have the appropriate permissions to access the course.

**group-membership-update** To add users to groups if you want to enroll a group.

**report-my-training** To list all courses and curriculums the current user is enrolled in, including the URL to access the course or curriculum.

**report-curriculum-taker** To get details of a user's progress within a curriculum.

**report-user-trainings-taken** To view the latest status of all of a user's courses and curriculums.

**report-user-training-transcripts** To list all of a user's transcripts and scores.

These actions work on courses, curriculums, and training folders and use the permissions allowed for objects in the Training library.

## Training library permissions

The Shared Training folder that you see in Adobe Connect Central is also called the Training library. Shared Training is called `courses` in the response from `sco-shortcuts`:

**Training**

```
<sco tree-id="123456" sco-id="123456" type="courses">
  <domain-name>example.com</domain-name>
</sco>
```

Each folder, course, and curriculum in the library is a SCO. As you navigate the Training library, you see the `sco-id` of the current course or curriculum in the browser URL. You can also retrieve the `sco-id` by calling `sco-contents` or `sco-expanded-contents` on a folder in the Training library.

Each course, curriculum, or content object in the Training library has permissions that define which users can access it. As you design your application, be aware of these permission levels:

**Enrollee permissions** Courses and curriculums have permissions that define which users are enrolled and can access them. The two permissions available are Enrolled and Denied.

**Training library permissions** Courses, curriculums, and folders in the Training library have either Manage or Denied permission. Manage permission means a user can create, delete, edit, or assign permissions. By default, users have Manage permission on their own training folders, and Administrators have Manage permission on any folder in the training library.

An Administrator can assign a user Manage permission on an individual course, curriculum, or folder with `permissions-update` or check the permissions a user has with `permissions-info`.

In XML API calls, you read, use, or set values of `permission-id` as you work with the Training library. These values of `permission-id` apply to courses and curriculums:

**view** The user has access to the course or curriculum, and permission is Enrolled.

**denied** The user is not allowed access, and permission is Denied.

You should also be aware of the permission a user has on a folder before executing an API call. Log in as a user with appropriate permission, or when needed, as your application's Administrator user. These values of `permission-id` apply to training folders:

**manage** The user can add, delete, change, or assign permissions to courses, curriculums, and content in a folder. The user can also list the contents of the folder with `sco-contents` or `sco-expanded-contents`.

**denied** The user cannot add, delete, change, or assign permissions to anything in the folder, but can list the contents of the folder.

## Find courses and curriculums

Most XML API actions that work with courses and curriculums require the `sco-id` of the course or curriculum. You often need to locate the `sco-id` dynamically, before you call another action, without knowing the exact name of the SCO.

Use these best practices to make searching for training SCOs efficient:

- Create specialized folders within the Shared Training folder for storing courses and curriculums. You can do this in Adobe Connect Central, or you can use the XML API, in which the Shared Training folder is named `courses`.
- Use these folders to store various categories of courses and curriculums, such as *Marketing Training* or *Sales Training*.
- Use a flat structure in the specialized folders, storing courses and curriculums one level deep.

This directory structure is also recommended when you want to display a list of all courses and curriculums (or all those in a subject area) and allow users to enroll themselves.

If you are working in Adobe Connect Central, you can find the `sco-id` of a course or curriculum by navigating to it, clicking its URL, and taking the value of `sco-id` from the browser URL. You can also locate the `sco-id` from an application, using the XML API.

### Find the `sco-id` of a course or curriculum

- 1 Call `sco-shortcuts`:

```
https://example.com/api/xml?action=sco-shortcuts
```

- 2 Parse the response for the `sco-id` of the `courses` folder:

```
<sco tree-id="624528" sco-id="624528" type="courses">  
  <domain-name>http://example.com</domain-name>  
</sco>
```

You cannot use a filter with `sco-shortcuts`, but you can parse the response for the `sco` element that has `type=courses`.

- 3 Call `sco-contents`, passing the `sco-id` of the `courses` folder and filtering for your specialized training folders:

```
https://example.com/api/xml?action=sco-contents&sco-id=624528  
&filter-name=Sales Training
```

- You can use `filter-name`, `filter-url-path`, another exact match filter, or a date filter. However, be careful when using `filter-like-name`, as it might affect server performance.
- You can also get the `sco-id` of your specialized training folder from the browser URL in Adobe Connect Central and pass it to `sco-contents`.

- 4 Parse the response for the `sco-id` of your specialized training folder:

```
<sco sco-id="2007122244" source-sco-id="" folder-id="624528"  
  type="folder" icon="folder" display-seq="0" is-folder="1">
```

- 5 Call `sco-contents`, passing it the `sco-id` of the specialized training folder and adding a filter that identifies the course or curriculum:

```
https://example.com/api/xml?action=sco-contents  
&sco-id=2007122244&filter-name=Java 201
```

- You can call `sco-contents`, rather than `sco-expanded-contents`, if all courses and curriculums are stored at the top level of your specialized training folder. This improves performance.
- You can define custom fields for SCOs if it helps you identify them in searches (see “[Create custom fields](#)” on page 22).

- 6 Parse the `sco` elements in the response for the `sco-id` of the course or curriculum:

```
<sco depth="2" sco-id="2006745673" folder-id="2006745671" type="content"  
  icon="course" lang="en" source-sco-id="2006744233"  
  display-seq="1" source-sco-type="0">  
  <name>All About Web Communities</name>  
  <url-path>/p33096345</url-path>  
  <description>Web 2.0 course</description>  
  <date-created>2006-06-12T14:48:25.870-07:00</date-created>  
  <date-modified>2006-06-12T14:48:25.870-07:00</date-modified>  
</sco>
```

### List all courses or curriculums available

- 1 Get the `sco-id` of a specialized training folder you have created.

You can also get the `sco-id` by navigating to the folder in Adobe Connect Central, clicking its URL, and reading the `sco-id` in the browser URL.

- 2 Call `sco-contents`, passing the folder's `sco-id`:

```
https://example.com/api/xml?action=sco-contents&sco-id=2006258748
```

The best practice is to create the specialized training folders one level deep. By doing so, you can call `sco-contents` rather than `sco-expanded-contents`. This gives better performance.

- 3 Parse the response for `name`, `url-path`, or any values you want to display:

```
<sco sco-id="2007035246" source-sco-id="2006334909"
      folder-id="2006258748" type="content" icon="course"
      display-seq="0" is-folder="0">
  <name>Java 101</name>
  <url-path>/java101/</url-path>
  <date-begin>2006-07-20T17:15:00.000-07:00</date-begin>
  <date-modified>2006-07-20T17:21:38.860-07:00</date-modified>
</sco>
```

## Create a course

You can use either Adobe Connect Central or Adobe Connect Web Services to create a course. If you use Web Services, first create an empty SCO and then add content to it.

- 1 Call `sco-update` to create a new SCO for the course:

```
https://example.com/api/xml?action=sco-update&name=salescourse&folder-
id=12345&icon=course&type=content
```

- 2 Parse the response for the `sco-id` value of the new course.
- 3 Add content to the new SCO, using the `sco-id` returned by `sco-update`:

```
https://example.com/api/xml?action=sco-update&sco-id=77711&source-sco-id=33444
```

- 4 Enroll users in the course (see [Enroll one user](#) and [Enroll a large number of users](#)).

## View a user's training

Once a user is logged in, you can list all courses the user is enrolled in with `report-my-courses`, or all of the user's courses and curriculums with `report-my-training`. This lists only the courses (or courses and curriculums) the user is enrolled in, not all courses available.

### View a user's courses and curriculums

- 1 Log the user in (see [Log in from an application](#)).

- 2 Call `report-my-training` to list all courses and curriculums the user is enrolled in:

```
https://example.com/api/xml?action=report-my-training
```

- 3 Parse the response for `name`, `url`, or any other values you want to display:

```
<row sco-id="2007035246" type="content" icon="course"
      permission-id="view">
  <name>Java 101</name>
  <url>example.com/java101/</url>
  <date-created>2006-07-20T17:21:11.940-07:00</date-created>
  <date-modified>2006-07-20T17:21:38.860-07:00</date-modified>
  <date-begin>2006-07-20T17:15:00.000-07:00</date-begin>
  <url-path>/java101/</url-path>
  <expired>>false</expired>
  <completed>>false</completed>
</row>
```

### View the status of all of the user's courses and curriculums

1 Get the `principal-id` of the user (see [Find a principal-id](#)).

2 Call `report-user-trainings-taken`:

```
https://example.com/api/xml?action=report-user-trainings-taken
&principal-id=2006258745
```

3 Parse the response for `status`:

```
<row transcript-id="2006293632" max-retries="" sco-id="2564016"
      type="content" icon="course" status="completed"
      certificate="2006293632" score="0" permission-id="" attempts="1">
  <name>Programming in Perl</name>
  <description>Info about Perl</description>
  <url-path>/p57283193/</url-path>
  <date-taken>2006-05-01T17:10:56.400-07:00</date-taken>
  <from-curriculum>false</from-curriculum>
</row>
```

A course can have many allowed values for `status`, but a curriculum can only have a status of `completed` or `incomplete`. The allowed values of `status` are described in [status attribute](#) in the reference.

## Enroll one user

To give users access to training, Adobe recommends that you enroll them in courses. This gives the users appropriate permission to launch and complete the course, and it gives you usage tracking and access to various report actions.

Courses differ from content. Courses are resumable and offer server-side review mode (for detailed information, see *Adobe Connect User Guide*).

Your application might allow users to self-enroll in courses, which involves calling `permissions-update` to enroll one user at a time. You may also want to write a workflow, which is a sequence of API calls, that creates a new user and enrolls the user in a course.

Enrolling users in training using the XML API (specifically, a call to `permissions-update`) does not send a notification. To send enrollment notifications, use Adobe Connect Central to enroll users.

### Enroll one user in a course or curriculum

1 Get the `sco-id` of the course (see [Find courses and curriculums](#)).

2 Get the `principal-id` of the user (see [List principals or guests](#)).

**Training**

- 3 To enroll the user in the course, call [permissions-update](#). Use the course `sco-id` as the `acl-id`, with a `permission-id` of `view`:

```
https://example.com/api/xml?action=permissions-update
    &acl-id=2007035246&principal-id=2006258745&permission-id=view
```

- 4 Call [report-my-training](#) to list all courses and curriculums the user is enrolled in:

```
https://example.com/api/xml?action=report-my-training
```

- 5 Parse the `row` elements in the response for values you want to display:

```
<row sco-id="2007035246" type="content" icon="course"
    permission-id="view">
  <name>Java 101</name>
  <url>example.com/java101/</url>
  <date-created>2006-07-20T17:21:11.940-07:00</date-created>
  <date-modified>2006-07-20T17:21:38.860-07:00</date-modified>
  <date-begin>2006-07-20T17:15:00.000-07:00</date-begin>
  <url-path>/java101/</url-path>
  <expired>>false</expired>
  <completed>>false</completed>
</row>
```

**Enroll a new user by workflow**

- 1 Call [principal-update](#) to create the new user and send a welcome e-mail:

```
https://example.com/api/xml?action=principal-update&first-name=jazz
    &last-name=doe&login=jazz@doe.com&password=hello&type=user
    &send-email=true&has-children=0&email=jazz@doe.com
```

To send the e-mail, make sure `send-email=true`.

- 2 Log the user in to the server:

```
https://example.com/api/xml?action=login&login=jazz@doe.com
    &password=hello&session=breezma6zor9rdfs8h6a
```

See [Log in from an application](#) for other ways to call `login`.

- 3 Call [group-membership-update](#) with `is-member=true` to add the user to the group:

```
https://example.com/api/xml?action=group-membership-update
    &group-id=4930296&principal-id=2006258745&is-member=true
```

- 4 Call [permissions-update](#) to enroll the user in a curriculum:

```
https://example.com/api/xml?action=permissions-update
    &acl-id=2006745669&principal-id=2007124930&permission-id=view
```

Use a `permission-id` of `view`.

- 5 Call [report-my-training](#) to list courses and curriculums the user is enrolled in:

```
https://example.com/api/xml?action=report-my-training
```

- 6 Parse the `row` elements in the response for values you want to display:



```
<row sco-id="2006745669" type="curriculum" icon="curriculum"
      permission-id="view">
  <name>A Day in the Life</name>
  <url>example.com/day/</url>
  <date-created>2006-06-12T14:47:59.903-07:00</date-created>
  <date-modified>2006-06-12T14:47:59.903-07:00</date-modified>
  <date-begin>2006-06-12T14:45:00.000-07:00</date-begin>
  <url-path>/day/</url-path>
  <expired>>false</expired>
  <completed>>false</completed>
</row>
```

## Enroll a large number of users

When you enroll a large number of users in a course, first decide whether to enroll the users directly or create a group and enroll it. Adobe recommends these best practices for enrolling users in courses:

- Enroll users directly in courses using `permissions-update`, which allows you to enroll 1000, 10,000, or more users with a single API call.
- Add the users to a group and enroll it only if you plan to reuse the group (for example, to enroll it in multiple courses). In this case, you can add only 200 users at a time.

### Enroll a large number of users (1000+) directly in a course

- 1 Get the `sco-id` of the course (see [Find courses and curriculums](#)).
- 2 Get the `principal-id` of each user you want to enroll.

To do this, you can:

- Call `principal-list` with filters to list the users you want to enroll:

```
https://example.com/api/xml?action=principal-list&filter-type=user
&filter-type=sales
```

- Read the values from a file.

- 3 Write a method that calls `permissions-update` with multiple trios of `acl-id`, `principal-id`, and `permission-id`:

```
https://example.com/api/xml?action=permissions-update
&acl-id=2007064258&principal-id=2007105030&permission-id=view&acl-
id=2007064258&principal-id=2006258745&permission-id=view ...
```

- The `acl-id` is the `sco-id` of the course.
- The `permission-id` is `view` to enroll users.
- The `principal-id` is unique in each trio.

If any trios have incorrect information, `permissions-update` returns an `ok` status, executes the correct trios, and does not execute the invalid ones.

- 4 Call `permissions-info` to check that the users have been enrolled:

```
https://example.com/api/xml?action=permissions-info
&acl-id=2007064258&filter-permission-id=view
```

Without a `principal-id`, this call returns a list of all principals enrolled in the course.

### Unenroll a large number of users (1000+) from a course

1 Get the `sco-id` of the course (see [Find courses and curriculums](#)).

2 Get the `principal-id` of each user you want to remove. You can:

- Call `principal-list` with filters to list the users you want to unenroll:

```
https://example.com/api/xml?action=principal-list&filter-type=user
&filter-account-id=624520
```

- Read the values from a file.

3 Write a method that calls `permissions-update` with multiple trios of `acl-id`, `principal-id`, and `permission-id`:

```
https://example.com/api/xml?action=permissions-update
&acl-id=2007064258&principal-id=2007105030&permission-id=denied&acl-
id=2007064258&principal-id=2006258745&permission-id=denied ...
```

The `permission-id` is `denied` to unenroll users from the course.

4 Call `permissions-info` to check that the users have been removed:

```
https://example.com/api/xml?action=permissions-info
&acl-id=2007064258&filter-permission-id=denied
```

### Enroll a large group (1000+) in a course

1 Create a group.

**With the XML API** Call `principal-update` and parse the response for the `principal-id`:

```
https://example.com/api/xml?action=principal-update&type=group
&has-children=1&name=developersc5
```

**With Adobe Connect** Central Create the group at Administration > Users and Groups > New Group. Take the `principal-id` of the new group from the browser URL.

2 Add the users you want to enroll to the group. You can use an API call or Adobe Connect Central, but you can add only 200 users at a time.

**With the XML API** Call `group-membership-update`, using multiple trios of `group-id`, `principal-id`, and `is-member=true`:

```
https://example.com/api/xml?action=group-membership-update
&group-id=4930296&principal-id=2006258745&is-member=true
&group-id=4930296&principal-id=2007343711&is-member=true
```

If any trios have incorrect information, `group-membership-update` returns an `ok` status, but the user in the incorrect trio is not added to the group.

**With Adobe Connect Central** Navigate to Administration > Users and Groups > Import. You can import users from a CSV (comma-delimited) file with at least a login ID for each user.

3 Get the `sco-id` of the course (see [Find courses and curriculums](#)) using the `sco-id` of the specialized training folder that contains the course.

4 Call `permissions-update` to enroll the group in the course:

```
https://example.com/api/xml?action=permissions-update
&acl-id=2007064258&principal-id=2007105030&permission-id=view
```

### Unenroll a large group (1000+) from a course

1 Call `permissions-info` on the course, filtering for a `permission-id` of `view`:

**Training**

```
https://example.com/api/xml?action=permissions-info
&acl-id=2007064258&filter-permission-id=view&filter-type=group
```

- 2 Parse the response for the `principal-id` of the group:

```
<principal principal-id="2006258745" is-primary="false" type="group"
  has-children="true" permission-id="view">
  <name>developers</name>
  <login>developers@acme.com</login>
</principal>
```

- 3 Call `permissions-update` with a `permission-id` of `denied` to remove the group's access to the course:

```
https://example.com/api/xml?action=permissions-update
&acl-id=2007064258&principal-id=2007105030&permission-id=denied
```

## View curriculum information

As training managers create curriculums and users take courses, you need to retrieve information about them to display in your application. Often you can make just a single call to get the information you need, once you have the `sco-id` of the curriculum or course and the user's `principal-id`.

You may, for example, want to display all users enrolled in a curriculum or all courses a curriculum has. Another common task is to display the courses in a curriculum the user has completed so far, and then display the remaining courses.

### Display all users enrolled in a course or curriculum

- 1 Call `permissions-info`, filtering for a `permission-id` of `view`:

```
https://example.com/api/xml?action=permissions-info
&acl-id=2006298444&filter-permission-id=view
```

- The `acl-id` is the `sco-id` of the course or curriculum.
- The `permission-id` of `view` means the user is enrolled.

- 2 Parse the response for `principal-id`, `name`, and any other values you need:

```
<principal principal-id="2006258745" is-primary="false" type="user"
  has-children="false" permission-id="view">
  <name>Joy Smith</name>
  <login>joy@acme.com</login>
</principal>
```

### Display a list of all training modules in a curriculum

A curriculum is a type of folder, and you can list its contents with `sco-contents` or `sco-expanded-contents`.

- 1 Get the `sco-id` of the curriculum (see [Find courses and curriculums](#)).
- 2 Call `sco-expanded-contents`, passing it the `sco-id`:

```
https://example.com/api/xml?action=sco-expanded-contents
&sco-id=2006745669
```

- 3 Parse the response for the `sco-id`, `folder-id`, and `depth`:

**Training**

```
<sco depth="1" sco-id="2006745674" folder-id="2006745669" type="link"
      icon="course" lang="en" source-sco-id="2006745673"
      display-seq="0" source-sco-type="0">
  <name>All About Web Communities</name>
  <url-path>/180422078/</url-path>
  <description>test</description>
  <date-created>2006-06-12T14:48:25.980-07:00</date-created>
  <date-modified>2006-06-12T14:48:25.980-07:00</date-modified>
</sco>
```

The response returns a flat list of `sco` elements, including the curriculum and each SCO it contains. You can build a hierarchy using the `sco-id`, `folder-id`, and `depth` values. The SCO with `type=curriculum` is the curriculum that contains the courses.

**View a user's completed and remaining work in a curriculum**

- 1 Get the `sco-id` of the curriculum (see [Find courses and curriculums](#)).
- 2 Get the `principal-id` of the user ([Find a principal-id](#)).
- 3 Call `report-curriculum-taker`, passing the `principal-id` as a `user-id`:

```
https://example.com/api/xml?action=report-curriculum-taker
&user-id=2006258745&sco-id=2006745669
```

- 4 Parse the response for the `status` attribute of each `sco` element and any other values you want to display in your application:

```
<sco transcript-id="2006745722" path-type="prereq-none" asset-id=""
      sco-id="2006745674" depth="1" folder-id="2006745669" type="15"
      icon="course" lang="en" max-retries="" source-sco-id="2006745673"
      source-sco-type="0" status="user-passed" score="0" certificate=""
      max-score="0" attempts="0">
  <access>access-open</access>
  <credit-granted>true</credit-granted>
  <name>All About Web Communities</name>
  <url-path>/180422078/</url-path>
  <description>test</description>
  <date-created>2006-06-12T15:06:02.947-07:00</date-created>
  <date-modified>2006-06-12T14:48:25.980-07:00</date-modified>
  <date-taken>2006-06-12T15:06:02.947-07:00</date-taken>
  <override>>false</override>
</sco>
```

- A status of `user-passed` or `completed` indicates a module the user has completed.
- A status of `not-attempted` or `incomplete` shows the user has not completed the module.
- The curriculum itself can only have a status of `completed` or `incomplete`.

## Report scores

Many courses offer learners a certain number of retries. If you use server-side review mode, a training manager can specify the maximum attempts the learner has to complete or pass the course successfully (see *Adobe Connect User Guide* for details of how course retry works in both server-side and client-side review mode).

This means that a learner can attempt a course multiple times and have multiple scores. In your application, you may want to display only the learner's highest score.

**Training****Report a user's highest score on a course or quiz**

- 1 Get the user's `principal-id` (see [List principals or guests](#)).
- 2 Get the `sco-id` of the course or quiz (see [Find courses and curriculums](#)).
- 3 Call `report-user-training-transcripts`, filtering on the `sco-id` and sorting on the score:

```
https://example.com/api/xml?action=report-user-training-transcripts
    &principal-id=2006258745&filter-sco-id=2006334909&sort-score=desc
```

- 4 Parse the response for the highest score, which should be in the first `row` element in the list:

```
<row transcript-id="2006335954" sco-id="2006334909"
    principal-id="2006258745" status="user-passed" score="20"
    max-score="20" certificate="2006335954" type="content"
    icon="producer">
    <name>Java Data Type Quiz</name>
    <url-path>/quiz/</url-path>
    <login>bob@acme.com</login>
    <date-taken>2006-05-12T11:55:24.940-07:00</date-taken>
    <principal-name>Bob Smith</principal-name>
</row>
```

# Chapter 7: Action reference

This section provides a reference for each action in the Adobe® Connect™ Web Services XML API.

All action (XML API), parameter, element, and attribute names are case sensitive. In other words, `name` is not the same as `Name`, and `sco-id` is not equivalent to `sco-ID`. You must enter them exactly as shown in this reference, unless a specific entry indicates an item is not case sensitive.

## What's new in Adobe Connect 8 Web Services

The following parameters are new in Adobe Connect 8 Web Services:

Action	Parameter value	Description
<a href="#">"meeting-feature-update"</a> on page 95	<code>fid-meeting-passcode-notallowed</code>	To disable the ability to passcode protect meeting rooms, call the action and pass this value for the <code>feature-id</code> parameter. For more information, see <a href="#">"Set or reset a meeting passcode"</a> on page 35.
<a href="#">"acl-field-update"</a> on page 64	<code>meeting-passcode</code>	To set or reset a meeting passcode, call the action and pass this value for the <code>field-id</code> parameter. For more information, see <a href="#">"Set or reset a meeting passcode"</a> on page 35.

A `report-meeting-session-users` action has been added.

This action provides information about all the user sessions for an Adobe Connect meeting session. A user session is created when a participant enters a meeting session. As more participants join the meeting, they join the meeting session. The user session ends when the user leaves the meeting session. When a new participant enters an empty meeting, a new meeting session and new user session is started.

For more information see ["report-meeting-session-users"](#) on page 141

## Sample action

### action name

#### Availability

The first version of Adobe Connect to support the action. Unless stated, the action is supported in all subsequent versions of Adobe Connect.

#### Description

A description of what the action does and when to use it.

#### Request URL

The syntax of an HTTP request URL.

**Action reference****Parameters**

A detailed description of the parameters in the request.

**Filters**

Specifies whether or not results can be filtered or sorted. For more information about filtering and sorting, see “[Filter and sort reference](#)” on page 214.

**Response structure**

The structure of an XML response.

**Response values**

A detailed description of the XML elements in a response.

**Sample request**

A sample HTTP request URL.

**Sample response**

A sample XML response.

**See also**

Links to related actions.

## Actions

### account-expiry-info

**Availability**

Acrobat Connect Pro Server 7

**Description**

Returns the expiration date of an account.

**Request URL**

```
http://server_name/api/xml
?action=account-expiry-info
&account-id=integer
&session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
account-id	Integer	N	The ID of the account for which you want expiration information. If you don't provide an account ID, the expiration date for the current user is returned.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Action reference**

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok"/>
  <Account account-id=integer>
    <name>String</name>
    <date-expired>Datetime</date-expired>
  </Account>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
Account		Container	Information about all fields describing the account.
	account-id	Integer	The ID of the account.
name		String	The name of the account.
date-expired		Datetime	The date the account expired.

**Sample request**

```
https://example.com/api/xml?action=account-expiry-info&account-id=7
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok"/>
  <Account account-id="7">
    <name>Enterprise Account</name>
    <date-expired>2009-09-11T18:15:00.000+05:30</date-expired>
  </Account>
</results>
```

**See also**

[expiry-settings-info](#), [expiry-settings-update](#)

**acl-field-info**

**Availability**

Breeze 5

**Description**

Returns information about a principal, account, or SCO, as defined in an access control list (ACL).

The returned information includes fields and their values. Each field has an ID—a name that describes the field.



**Action reference**

To call `acl-field-info`, you must have view permission for the principal, account, or object. You must also specify a value for `acl-id`, which is the object the principal has access to. The `acl-id` can be a `sco-id`, an `account-id`, or a `principal-id`. You can call `principal-list` to determine the `account-id` or `principal-id`, or `sco-shortcuts` or `sco-contents` to get a `sco-id`.

**Request URL**

```
http://server_name/api/xml
  ?action=acl-field-info
  &acl-id=integer
  &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
<code>acl-id</code>	Integer	Y	The ID of the SCO, account, or principal for which you want field information. Can be a valid <code>sco-id</code> , <code>account-id</code> , or <code>principal-id</code> .
<code>session</code>	String	N	The value of the <code>BREEZESSESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <acl-fields>
    <field acl-id=integer field-id=string>
      <value>string</value>
    </field>
    ...
  </acl-fields>
</results>
```

**Response values**

Element	Attribute	Type	Description
<code>results</code>		Container	All results the action returns.
<code>status</code>		Empty, with attributes	The status of the response.
	<code>code</code>	Allowed value	A code indicating the response status (see <code>status</code> ).
<code>acl-fields</code>		Container	Information about all fields describing the principal, account, or object.
<code>field</code>		Container	One field describing the principal, account, or object.
	<code>acl-id</code>	Integer	The <code>acl-id</code> specified in the request, which is a <code>sco-id</code> , <code>principal-id</code> , or <code>account-id</code> .
	<code>field-id</code>	String	The name of the field.
<code>value</code>		String	The value of the field.

**Action reference****Sample request**

```
https://example.com/api/xml?action=acl-field-info&acl-id=2006258745
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <acl-fields>
    <field acl-id="2006258745" field-id="email">
      <value>joy@acme.com</value>
    </field>
    <field acl-id="2006258745" field-id="first-name">
      <value>Joy</value>
    </field>
    <field acl-id="2006258745" field-id="last-name">
      <value>Smith</value>
    </field>
  </acl-fields>
</results>
```

**See also**

[acl-field-list](#), [acl-field-update](#)

## acl-field-list

**Availability**

Breeze 5

**Description**

Returns a list of values for all instances of a field name on your Adobe Connect Server account.

For example, to list the first names of all users in the account, call `acl-field-list` with `field-id=first-name`.

You can call [acl-field-info](#) first to get a list of field names.

**Request URL**

```
http://server_name/api/xml
  ?action=acl-field-list
  &field-id=string
  &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
field-id	String	Y	The name of a field in the access control list for which you want values and IDs. Only one field name is allowed.
session	String	N	The value of the BREEZESESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Action reference**

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <acl-field-list>
    <acl acl-id=integer>
      <value>string</value>
    </acl>
    ...
  </acl-field-list>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
acl-field-list		Container	Information about all of the values in the account for the specified field.
acl		Container	Information about one value for the specified field.
	acl-id	Integer	The ID of the principal, SCO, or account the field belongs to.
value		String	The value of the field.

**Sample request**

```
https://example.com/api/xml?action=acl-field-list&field-id=first-name
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok"/>
  <acl-field-list>
    <acl acl-id="381223">
      <value>John</value>
    </acl>
    <acl acl-id="381302">
      <value>Daryl</value>
    </acl>
    <acl acl-id="381405">
      <value>Mary</value>
    </acl>
  </acl-field-list>
</results>
```

**See also**

[acl-field-info](#), [acl-field-update](#)

## acl-field-update

### Availability

Breeze 5

### Description

Updates the value of an ACL field that belongs to a SCO or an account.

**Note:** To update a standard field for a principal (a user or a group), use the *principal-update* action. To update a custom field for a principal, use the *acl-field-update* action.

Each SCO or account belongs to at least one access control list (ACL). The ACL lists the principals that have permission to access the SCO or account.

Call [acl-field-info](#) to determine the fields in the ACL for a SCO or account. The response contains the `field-id` you need for the request to `acl-field-update`:

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <acl-fields>
    <field acl-id="2006258745" field-id="email">
      <value>joy@acme.com</value>
    </field> ...
  </acl-fields>
</results>
```

You can specify multiple trios of `acl-id`, `field-id`, and `value`. If you do, use an HTTP `POST` method, rather than a `GET`, to make the request. The `GET` method has limitations that might cause the request to be truncated. With a `POST`, you can add about 50 trios to the request.

To call `acl-field-update`, you need `modify` permission on the SCO or account.

### Request URL

```
http://server_name/api/xml
?action=acl-field-update
&acl-id=integer
&field-id=string
&value=string
&session=BreezeSessionCookieValue
```

### Parameters

Name	Type	Required	Description
<code>acl-id</code>	Integer	Y	The ID of the SCO or account. Can be a valid <code>sco-id</code> or <code>account-id</code> .
<code>field-id</code>	String	Y	The name of the field for which you want to update value. The field can be a server-defined field or a custom field. A custom field has a <code>field-id</code> starting with <code>x-</code> , such as <code>x-12056</code> .
<code>value</code>	String	Y	The value to set.
<code>session</code>	String	N	The value of the <code>BREEZESESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Action reference****Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Sample request**

```
https://example.com/api/xml?action=acl-field-update&acl-id=2007035246
  &field-id=name&value=Java 101
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
</results>
```

**See also**

[acl-field-list](#)

## acl-preference-update

**Availability**

Breeze 4

**Description**

Updates a user profile with new language and time zone settings.

**Request URL**

```
http://server_name/api/xml
  ?action=acl-preference-update
  &acl-id=integer
  &lang=allowedValue
  &time-zone-id=allowedValue
  &session=BreezeSessionCookieValue
```

**Action reference****Parameters**

Name	Type	Required	Description
acl-id	Integer	Y	The ID of the user whose preferences will be updated. Can be a valid <code>principal-id</code> .
lang	Allowed value	N	An abbreviation for the new language (see <a href="#">lang</a> for valid values).
time-zone-id	Allowed value	N	An integer setting for the new time zone (see <a href="#">time-zone-id</a> for values).
session	String	N	The value of the <code>BREEZESESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
<status code=allowedValue />
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Sample request**

```
https://server.com/api/xml?action=acl-preference-update&acl-id=12345
&lang=fr&time-zone-id=0
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
</results>
```

**common-info****Availability**

Breeze 4

**Description**

Returns basic information about the current user and the Adobe Connect server or Adobe Connect hosted account, including the value of the `BREEZESESSION` cookie.

If you call `common-info` without logging in, the response does not contain `user` and `account` elements, because the server cannot identify a user. However, even without logging in, `common-info` returns a `BREEZESESSION` cookie value.

**Action reference**

The response also contains `host`, `local-host`, and `admin-host` elements. If Adobe Connect is hosted on a cluster, `host` is the cluster name; `local-host` is the name of the server in the cluster that executes the call to `common-info`; and `admin-host` is the name of the secure host on a cluster that supports SSL. Your application can use the value of `admin-host` to convert HTTP URLs to more secure HTTPS URLs.

**Request URL**

```
http://server_name/api/xml
    ?action=common-info
    &domain=string
    &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
domain	String	N	A domain name identifying a Adobe Connect hosted account. Use to get information about your hosted account.
session	String	N	The value of the BREEZESSESSION cookie . Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <common locale=allowedValue time-zone-id=integer>
    <cookie>string</cookie>
    <date>datetime</date>
    <host>url</host>
    <local-host>hostname</local-host>
    <admin-host>hostname</admin-host>
    <url>/api/xml?action=common-info</url>
    <version>string</version>
    <account account-id=integer />
    <user user-id=integer type="user">
      <name>string</name>
      <login>string</login>
    </user>
    <user-agent>string</user-agent>
  </common>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Action reference**

Element	Attribute	Type	Description
common		Container	Common information about your connection to the server.
	locale	Allowed value	A setting that defines how Connect Central or your application displays information to a user (see <a href="#">lang</a> for values).
	time-zone-id	Allowed value	A code that defines the user's time zone (see <a href="#">time-zone-id</a> for values).
cookie		String	The value of the BREEZESSESSION cookie (a string the server returns identifying this user for this login session).
date		Datetime	The date and time the call to <code>common-info</code> was made, in <a href="#">ISO 8601</a> format.
host		String	If Adobe Connect runs on a server, the URL of the fully qualified host name of the server. If a cluster, the name that identifies the cluster.
local-host		String	The name of the computer that executed the action (on a single server, the same as <code>host</code> ; on a cluster, the name of the server that executed the action).
admin-host		String	The name of the secure host on a cluster that supports SSL.
url		String	The part of the URL making this call that identifies the action name.
version		String	The server version name and number.
account		Empty, with attribute	Information about the account the user belongs to. Returned if you are logged in to Adobe Connect or are making the call on a Adobe Connect hosted account.
	account-id	Integer	The ID of the account the user belongs to.
user		Container	Information about the user who established a session with the server. Returned only if the user making the call is logged in.
	user-id	Integer	The ID of the user who established a session with the server.
	type	Allowed value	The type of principal who has a session (usually <code>user</code> ; see allowed values for principals at <a href="#">type</a> ).
name		String	The full name of the user who established a session with the server.
login		String	The login name of the user who is logged in to the server, often the user's e-mail address.
user-agent		String	The identifier of the web browser or client that established a session with the server.

**Sample request**

`https://example.com/api/xml?action=common-info`



**Action reference**

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <common locale="en" time-zone-id="4">
    <cookie>breezsi4dundh5srw2fq6</cookie>
    <date>2006-09-08T11:17:04.470-07:00</date>
    <host>https:example.com</host>
    <local-host>localserver17</local-host>
    <admin-host>securehost.com</admin-host>
    <url>/api/xml?action=common-info</url>
    <version>connect_6000</version>
    <account account-id="624520" />
    <user user-id="2006258745" type="user">
      <name>Joy Smith</name>
      <login>joy@acme.com</login>
    </user>
    <user-agent>Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1;
      .NET CLR 1.1.4322)</user-agent>
  </common>
</results>
```

**curriculum-contents**

**Availability**

Connect Pro 7

**Description**

Lists all of the SCOs in a curriculum, including the contents of subfolders.

*Note: To list the contents of a curriculum, use this action instead of `sco-expanded-contents`*

To find a `sco-id` to pass in the request URL, call [sco-shortcuts](#). For more information, see “[Find SCOs](#)” on page 24.

**Request URL**

```
http://server_name/api/xml
  ?action=curriculum-contents
  &sco-id=integer
  &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
sco-id	Integer	Y	The unique ID of a curriculum.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You cannot sort or filter the response to this API call.

**Action reference**

**Response structure**

```
<results>
  <status code=allowedValue/>
  <curriculum-contents>
    <sco depth=integer sco-id=integer folder-id=integer type=allowedValue
icon=allowedValue lang=allowedValue source-sco-id=integer display-seq=integer source-sco-
type=integer source-sco-icon=integer content-source-sco-icon=integer>
      <name>string</name>
      <url-path>string</url-path>
      <description>
        string
      </description>
      <date-created>datetime</date-created>
      <date-modified>datetime</date-modified>
    </sco>
    ...more sco elements...
  </curriculum-contents>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
curriculum-contents		Container	The list of all SCOs the curriculum contains.
sco		Container	Details about one SCO. This SCO can be a folder or any other type of object.
	depth	Integer	The depth in the content tree at which this object appears, with top-level objects at 1.
	sco-id	Integer	The unique ID of the SCO. If the SCO is a folder, same as <code>folder-id</code> .
	folder-id	Integer	The ID of the folder the SCO belongs to.
	type	Allowed value	The type of this content object (see <a href="#">type</a> ).
	icon	Allowed value	The name of the icon that visually identifies this object.
	lang	Allowed value	The language in which information about the SCO is displayed (see <a href="#">lang</a> for values).
	source-sco-id	Integer	The ID of a SCO from which this SCO was created, such as a meeting template or course content.
	display-seq	Integer	The sequence in which Connect Central (or your application, if you use this value) displays a list of SCOs. Values are not necessarily unique, so multiple SCOs can have the same <code>display-seq</code> value. In that case, the application must define the display sequence. The default is 0.
	source-sco-type	Integer	An integer indicating the type of SCO from which this SCO was created.
	source-sco-icon	Integer	An integer indicating the type of icon from which this icon was created.

**Action reference**

Element	Attribute	Type	Description
	content-source-sco-icon	Integer	An integer indicating the type of content from which this icon was created.
name		String	The name of the contained SCO.
url-path		String	The URL of the SCO within the curriculum.
description		String	The summary in the UI. If a SCO has a summary, this field exists, otherwise, the field does not exist.
date-created		Datetime	The date and time the principal began interacting with the SCO and the transaction was created.
date-modified		Datetime	The date the SCO was last updated.

**Sample request**

http://example.com/api/xml?action=curriculum-contents&sco-id=11697&session=breezq7dyhc7m3de8dkrsr

**Sample response**

```
<results>
  <status code="ok"/>
  <curriculum-contents>
    <sco depth="1" sco-id="31949" folder-id="11697" type="link" icon="course" lang="en"
source-sco-id="41184" display-seq="0" source-sco-type="0" source-sco-icon="1" content-source-
sco-icon="1025">
      <name>FlashBelt09</name>
      <url-path>/166176109/</url-path>
      <date-created>2009-05-27T03:48:54.277+05:30</date-created>
      <date-modified>2009-05-27T03:48:54.277+05:30</date-modified>
    </sco>
    <sco depth="0" sco-id="11697" folder-id="41177" type="curriculum" icon="curriculum"
lang="en" source-sco-id="" display-seq="0" source-sco-type="" source-sco-icon="" content-
source-sco-icon="">
      <name>Backyard Cooking</name>
      <url-path>/cooking/</url-path>
      <description>
        Learn how to cook with things you can find in most urban backyards.
      </description>
      <date-begin>2009-05-27T03:45:00.000+05:30</date-begin>
      <date-created>2009-05-27T03:48:40.383+05:30</date-created>
      <date-modified>2009-05-27T03:48:40.383+05:30</date-modified>
    </sco>
  </curriculum-contents>
</results>
```

**custom-fields**

**Availability**

Breeze 4

**Description**

Lists all custom fields defined in an account and details about the fields.

**Action reference**

Custom fields provide information about objects (SCOs) or principals that is not already defined in Connect Central. You can create custom fields, or update their value, using [custom-field-update](#).

**Request URL**

```
http://server_name/api/xml
    ?action=custom-fields
    &filter-definition=value
    &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
<a href="#">filter-definition</a>	Filter definition	N	A filter to reduce the volume of the response.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter the response on any element or attribute it contains.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <custom-fields>
    <field permission-id=allowedValue object-type=allowedValue
      field-id=string account-id=integer display-seq=integer
      field-type=allowedValue is-primary=boolean is-required=boolean>
      <name>string</name>
    </field>
  </custom-fields>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
custom-fields		Container	The list of custom fields that match the query.
field		Container	Details about one custom field.
	permission-id	Allowed value	The permission the current user has to access the custom field (see <a href="#">permission-id</a> for values).
	object-type	Allowed value	The type of object the custom field describes (see <a href="#">permission-id</a> ).
	field-id	String	The name of the field, as identified on the server.
	account-id	Integer	The ID of the account in which the custom field is defined.

**Action reference**

Element	Attribute	Type	Description
	display-seq	Integer	The sequence in which Connect Central or your application displays the custom field, relative to other custom fields.
	field-type	Allowed value	The type of data the custom field accepts. Allowed values are <code>text</code> , <code>textarea</code> , and <code>password</code> .
	is-primary	Boolean	Whether the custom field can be deleted ( <code>true</code> if no, and <code>false</code> if yes).
	is-required	Boolean	Whether this custom field is required. <code>true</code> if a value must be specified for this field in each object that uses it. Otherwise, <code>false</code> .
name		String	The name of the custom field as Connect Central or your application displays it.

**Sample request**

```
https://example.com/api/xml?action=custom-fields&filter-like-name=name
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <custom-fields>
    <field permission-id="manage" object-type="object-type-principal"
      field-id="first-name" account-id="624520" display-seq="1"
      field-type="text" is-primary="true" is-required="true">
      <name>First Name</name>
    </field>
    <field permission-id="manage" object-type="object-type-principal"
      field-id="last-name" account-id="624520" display-seq="2"
      field-type="text" is-primary="true" is-required="true">
      <name>Last Name</name>
    </field>
  </custom-fields>
</results>
```

**See also**

[custom-field-update](#)

**custom-fields-delete****Availability**

Breeze 4

**Description**

Deletes a custom field.

The value of `is-primary` for a custom field must be `false` before the field can be deleted. If `is-primary` is `true` and you want to change its value, call [custom-field-update](#).

**Action reference****Request URL**

```
http://server_name/api/xml
  ?action=custom-fields-delete
  &field-id=string
  &object-type=allowedValue
  &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
field-id	String	Y	The ID of the field to be deleted. Call <code>custom-fields-delete</code> to obtain the ID, which is returned in the <code>field-id</code> attribute of the <code>field</code> element.
object-type	String	Y	The type of SCO for which the field is defined (for values, see <a href="#">type</a> ).
session	String	N	The value of the <code>BREEZESESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Sample request**

```
https://example.com/api/xml?action=custom-fields-delete&field-id=2006338719&object-
type=object-type-principal
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
</results>
```

**See also**

[custom-field-update](#)

## custom-field-update

### Availability

Breeze 4

### Description

Creates a new custom field or updates the value of an existing one.

You can define up to eight custom fields on a principal or SCO. To create a custom field, call `custom-field-update` with at least the following fields: `object-type`, `permission-id`, `name`, `field-type`, `is-required`, and `is-primary`. If `custom-field-update` is successful, it returns a `field-id`.

To update a custom field, specify the `field-id`, an `object-type`, and a name for each field that has a value you want to change.

Be careful when defining custom fields, as retrieving those fields in a report (for example, by calling `report-bulk-users`) can affect the performance of the server and the database.

### Request URL

```
http://server_name/api/xml
?action=custom-field-update
&account-id=integer
&object-type=object-type-allowedValue
&permission-id=allowedValue
&name=string
&comments=string
&field-type=allowedValue
&is-required=boolean
&is-primary=boolean
&display-seq=integer
&field-id=integer
&session=BreezeSessionCookieValue
```

**Action reference**

**Parameters**

Name	Type	Required	Description
account-id	Integer	N	The account ID in which the field is created.
object-type	String	Y	The type of SCO this field applies to. Required to create and update fields. Allowed values: <ul style="list-style-type: none"> <li>• object-type-principal</li> <li>• object-type-meeting</li> <li>• object-type-sco</li> <li>• object-type-event</li> <li>• object-type-read-only</li> </ul> Example: object-type=object-type-principal The value <code>object-type-read-only</code> means that Connect Central displays the value but a user cannot set it using Connect Central. You can also use this value in custom applications.
permission-id	String	Y	The permission a principal needs on the object to set or view the field's value. The only allowed value is <code>manage</code> . Required to create a field.
name	String	Y	The label for the field in the user interface. Required to create a field.
comments	String	N	Any comments you define for the custom field, displayed as hint text in your user interface. Can be up to 60 characters long.
field-type	String	Y	The type of field. Allowed values are <code>text</code> , <code>textarea</code> , and <code>password</code> . Required to create a field.
is-required	Boolean	Y	Whether this custom field is required. Use <code>true</code> if a value must be specified for this field in each object that uses it. Otherwise, use <code>false</code> . Required to create a field.
is-primary	Boolean	Y	Whether this custom field can be deleted through the user interface ( <code>true</code> if it cannot be deleted, and <code>false</code> if it can).
display-seq	Integer	N	The sequence in which Connect Central or your application displays the custom field, relative to other custom fields.
field-id	Integer	Y	The name of a field that has a value you want to update. Required to update a field.
session	String	N	The value of the <code>BREEZESESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.



**Action reference**

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <field field-id=integer display-seq=integer object-type=allowedValue
    account-id=integer is-primary=boolean permission-id=allowedValue
    is-required=boolean field-type=string>
    <comments>string</comments>
    <name>string</name>
  </field>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
field		Empty, with attributes	Information about the custom field.
	field-id	Integer	A numeric identifier for the field.
	display-seq	Integer	The sequence in which Connect Central or your application displays the field.
	object-type	Allowed value	The type of object the field describes (see <a href="#">type</a> for allowed values).
	account-id	Integer	For customers on Adobe Connect hosted accounts, the ID of the account in which the field is defined.
	is-primary	Boolean	Whether this custom field can be deleted ( <code>true</code> if no, <code>false</code> if yes).
	permission-id	Allowed value	The permission needed to access the custom field (see <a href="#">permission-id</a> ).
	is-required	Boolean	Whether a value for this custom field is required ( <code>true</code> if yes and <code>false</code> if no).
	field-type	Allowed value	The type of data the field accepts. Allowed values are <code>text</code> , <code>textarea</code> , and <code>password</code> .
comments		String	The comment entered in <code>comments</code> in the request.
name		String	The name of the field entered in <code>name</code> in the request.

**Sample request**

```
https://example.com/api/xml?action=custom-field-update
&object-type=object-type-principal&permission-id=manage
&account-id=624520&name=jobtitle&comments=test&field-type=text
&is-required=true&is-primary=false&display-seq=1
```

**Action reference**

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <field field-id="2006472106" object-type="object-type-principal"
    display-seq="1" account-id="624520" is-primary="false"
    permission-id="manage" is-required="true" field-type="text">
    <comments>test</comments>
    <name>jobtitle</name>
  </field>
</results>
```

**See also**

[report-bulk-users](#)

## expiry-settings-info

**Availability**

Acrobat Connect Pro Server 7

**Description**

Returns information about the current settings for account-expiration notifications (the warnings given to users before an account expires). A user is notified *x* number of days before their account expires. This action simply returns the value of *x*.

**Request URL**

```
https://example.com/api/xml
  ?action=expiry-settings-info
  &account-id=Integer
  &session=String
```

**Parameters**

Name	Type	Required	Description
account-id	Integer	N	The ID of the account. If you don't provide an account ID, the information for the current account is returned.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <expiry-num-of-days>
    <value>30</value>
  </expiry-num-of-days>
</results>
```

**Action reference****Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
expiry-num-of-days		Container	Information about the current settings for account-expiration notifications.
value		Integer	The user is notified this many days before their account expires. The default value is 30. For example, if a user's account expires on December 31, the user is notified on December 1.

**Sample request**

```
https://example.com/api/xml?action=expiry-settings-info&account-id=7
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok"/>
  <expiry-num-of-days>
    <value>30</value>
  </expiry-num-of-days>
</results>
```

**See also**

[expiry-settings-update](#)

**expiry-settings-update****Availability**

Acrobat Connect Pro Server 7

**Description**

Updates information about the settings for account-expiration notification (the notification given to users before an account expires). A user is notified *x* number of days before their account expires. This action simply updates the value of *x*.

**Request URL**

```
https://example.com/api/xml
?action=expiry-settings-update
&account-id=Integer
&session=String
```

**Action reference**

**Parameters**

Name	Type	Required	Description
account-id	Integer	N	The ID of the account. If you don't provide an account ID, the information for the current account is updated.
expiry-num-of-days	Integer	Y	A user is notified this many days before their account expires. The default value is 30; possible values are 30, 60, and 90.  For example, if the value of this parameter is 30, a user is notified 30 days before their account is due to expire.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Sample request**

https://example.com/api/xml?action=expiry-settings-update&account-id=7&expiry-num-of-days=30

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok"/>
</results>
```

**See also**

[account-expiry-info](#), [expiry-settings-info](#)

## group-membership-update

**Availability**

Breeze 4

**Description**

Adds one or more principals to a group, or removes one or more principals from a group.

To update multiple principals and groups, specify multiple trios of `group-id`, `principal-id`, and `is-member` parameters.

**Action reference**

You can obtain a `group-id` by calling [principal-list](#) and filtering the response with `filter-type=group` or another filter value such as `filter-type=admins`. The built-in groups have distinctive types other than `group` (see [type](#) for a list of values).

**Request URL**

```
http://server_name/api/xml
  ?action=group-membership-update
  &group-id=integer
  &principal-id=integer
  &is-member=boolean
  &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
group-id	Integer	Y	The ID of the group in which you want to add or change members.
principal-id	Integer	Y	The ID of the principal whose membership status you want to update. Returned by <code>principal-info</code> .
is-member	Boolean	Y	Whether the principal is added to ( <code>true</code> ) or deleted from ( <code>false</code> ) the group.
session	String	N	The value of the <code>BREEZESSESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	Top-level element for the response.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Sample request**

```
https://example.com/api/xml?action=group-membership-update&group-id=632398
  &principal-id=2006258745&is-member=true
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
</results>
```

## learning-path-info

### Availability

Breeze 5

### Description

Returns a list of learning paths for a learning object that belongs to a curriculum.

A learning object is any SCO that has been added to a curriculum. A learning path is determined by rules that establish whether a learner can proceed to the next learning object.

You can create a learning path by establishing prerequisite requirements, completion requirements, or preassessment requirements. For example, a learning path might be the rule that the class *Welcome to AcmeCo* must be completed before *Managing Projects at AcmeCo*.

A call to `learning-path-info` lists modules within a curriculum and their paths to each other. To see the complete contents of a curriculum, including content, meetings, and so on, call `sco-expanded-contents`.

### Request URL

```
http://server_name/api/xml
  ?action=learning-path-info
    &curriculum-id=integer
    &sco-id=integer
    &filter-definition=value
    &sort-definition=value
    &session=BreezeSessionCookieValue
```

### Parameters

Name	Type	Required	Description
<code>curriculum-id</code>	Integer	Y	The ID of the curriculum the learning object belongs to.
<code>sco-id</code>	Integer	Y	The ID of the curriculum module (course, presentation, or similar) for which you want a learning path.
<code>filter-definition</code>	Filter definition	N	A filter to reduce the volume of the response.
<code>sort-definition</code>	Sort definition	N	A sort to return results in a certain sequence.
<code>session</code>	String	N	The value of the <code>BREEZESSESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

### Filters

You can filter or sort the response on any element or attribute it contains.

**Action reference****Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <learning-paths>
    <learning-path curriculum-id=integer current-sco-id=integer target-sco-id=integer
path-type=allowedValue>
      <name>string</name>
    </learning-path>
  </learning-paths>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
learning-paths		Container	Information about learning paths in a curriculum.
	curriculum-id	Integer	The numeric ID of the curriculum.
	current-sco-id	Integer	The learning object for which you want a path.
	target-sco-id	Integer	The ID of the learning object that restricts access to the current learning object (for example, a prerequisite learning object).
	path-type	Allowed value	The type of path between the target and current learning objects (for example, whether completion of the target is required as a prerequisite). See <a href="#">path-type</a> for allowed values.
name		String	The name of the target learning object.

**Sample request**

```
https://example.com/api/xml?action=learning-path-info&sco-id=2006334909
&curriculum-id=2006298444
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <learning-paths>
    <learning-path curriculum-id="2006298444" current-sco-id="2006298444"
target-sco-id="2006298445" path-type="completion-required">
      <name>Security at AcmeCo</name>
    </learning-path>
  </learning-paths>
</results>
```

**See also**

[learning-path-update](#)

## learning-path-update

### Availability

Breeze 5

### Description

Updates the learning path for a single learning object in a curriculum. A learning object is any SCO that is added to a curriculum.

### Request URL

```
http://server_name/api/xml
?action=learning-path-update
&curriculum-id=integer
&current-sco-id=integer
&target-sco-id=integer
&path-type=allowedValue
&session=BreezeSessionCookieValue
```

### Parameters

Name	Type	Required	Description
curriculum-id	Integer	Y	The ID of the curriculum to which this learning object belongs.
current-sco-id	Integer	N	The ID of the learning object that has the access you want to update.
target-sco-id	Integer	N	The ID of the learning object that restricts access to the current learning object (for example, a prerequisite course).
path-type	Allowed value	Y	The type of path between the target learning object and the current learning object (see <a href="#">path-type</a> for allowed values).
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

### Filters

Results cannot be filtered or sorted.

### Response structure

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
</results>
```

### Response values

Element	Attribute	Type	Description
results		Container	Top-level element for the response.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).



**Action reference****Sample request**

```
https://example.com/api/xml?action=learning-path-update
&curriculum-id=2006298444&current-sco-id=2007064258
&target-sco-id=2007035246&path-type=completion-required
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok"/>
</results>
```

**See also**

[learning-path-info](#)

**limited-administrator-permissions info****Availability**

Acrobat Connect Pro 7

**Description**

Returns a list of permissions that can be enabled or disabled for the Limited Administrators group and whether or not that permission is currently enabled. For more information on Limited Administrators, see [limited-administrator-permissions-update](#).

**Request URL**

```
http://server_name/api/xml
?action=limited-administrator-permissions-info
&session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8"?>
<results>
  <status code="ok"/>
  <permissions>
    <permission>
      <enabled>Boolean</enabled>
      <name>string</name>
    </permission>
  </permissions>
</results>
```

**Action reference****Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
permissions		Container	A list of permissions.
permission		Container	A list of information about the permission.
enabled		Boolean	A value indicating whether the permission is enabled ( <code>true</code> ) or not ( <code>false</code> ).
name		String	The name of the permission.

**Sample request**

```
https://example.com/api/xml?action=limited-administrator-permissions-info
&session=breeze6qdeheiso93efb5
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8"?>
<results>
  <status code="ok"/>
  <permissions>
    <permission>
      <enabled>true</enabled>
      <name>edit-account-info</name>
    </permission>
    <permission>
      <enabled>>false</enabled>
      <name>view-disk-usage-and-reports</name>
    </permission>
    <permission>
      <enabled>true</enabled>
      <name>reset-password</name>
    </permission>
    <permission>
      <enabled>true</enabled>
      <name>view-user-data</name>
    </permission>
    <permission>
      <enabled>true</enabled>
      <name>add-users-groups-webui</name>
    </permission>
    <permission>
      <enabled>>false</enabled>
      <name>add-users-groups-csv</name>
    </permission>
    <permission>
      <enabled>true</enabled>
      <name>set-content-meeting-permissions</name>
    </permission>
    <permission>
      <enabled>true</enabled>
      <name>user-profile-fields</name>
    </permission>
  </permissions>
</results>
```

**Action reference**

```

    </permission>
  <permission>
    <enabled>true</enabled>
    <name>change-login-pw-policy</name>
  </permission>
  <permission>
    <enabled>>false</enabled>
    <name>delete-users-groups</name>
  </permission>
  <permission>
    <enabled>true</enabled>
    <name>modify-current-users-groups</name>
  </permission>
  <permission>
    <enabled>>false</enabled>
    <name>customization</name>
  </permission>
  <permission>
    <enabled>>false</enabled>
    <name>compliance</name>
  </permission>
  <permission>
    <enabled>>false</enabled>
    <name>chargebacks</name>
  </permission>
  <permission>
    <enabled>>false</enabled>
    <name>view-system-usage-reports</name>
  </permission>
  <permission>
    <enabled>>false</enabled>
    <name>quota-threshold-notifications</name>
  </permission>
</permissions>
</results>

```

**limited-administrator-permissions-update****Availability**

Acrobat Connect Pro 7

**Description**

Updates the permissions that can be enabled for Limited Administrators.

With Limited Administrators, your organization can have finer control over administrators and what types of things they can access. Your organization can separate system administrators who control all aspects of the system from Limited Administrators, who can access and control a subset of the system.

Each Adobe Connect installation has one Limited Administrators group. Users in the Administrators group can edit the permissions of Limited Administrators.

**Action reference**

**Request URL**

```
http://server_name/api/xml
?action=limited-administrator-permissions-update
&view-disk-usage-and-reports=boolean
&reset-password=boolean
&view-user-data=boolean
&add-users-groups-webui=boolean
&add-users-groups-csv=boolean
&user-profile-fields=boolean
&change-login-pw-policy=boolean
&delete-users-groups=boolean
&modify-current-users-groups=boolean
&customization=boolean
&edit-account-info=boolean
&set-content-meeting-permissions=boolean
&compliance=boolean
&chargebacks=boolean
&view-training-reports=boolean
&reset-to-default=value
```

**Parameters**

When you use this command, pass at least one parameter. The descriptions that follow indicate if the permission is set to true by default.

Name	Type	Required	Description
view-disk-usage-and-reports	Boolean	N	A value of true allows limited administrators to view disk usage and reports. The default value is true.
reset-password	Boolean	N	A value of true allows limited administrators to reset the password of a user. Part of the view-user-data set. The default value is true.
view-user-data	Boolean	N	Superset; a value of true allows limited administrators to view user data. By setting this parameter to enable, you enable all parameters in this set. (See all parameters that are part of the view-user-data set.) The default value is true.
add-users-groups-webui	Boolean	N	A value of true allows limited administrators to add users and groups by using the management console. Part of the view-user-data set. The default value is true.
add-users-groups-csv	Boolean	N	A value of true allows limited administrators to add users or groups by importing a CSV file. Part of the view-user-data set
user-profile-fields	Boolean	N	A value of true allows limited administrators to modify user profile fields.
change-login-pw-policy	Boolean	N	A value of true allows limited administrators to change the login and password policies.
delete-users-groups	Boolean	N	A value of true allows limited administrators to delete users or groups. Part of the view-user-data set
modify-current-users-groups	Boolean	N	A value of true allows limited administrators to modify currents users and groups. Part of the view-user-data set. The default value is true.
customization	Boolean	N	A value of true allows limited administrators to customize the colors of the account web pages, meetings, and the login page.
edit-account-info	Boolean	N	A value of true allows limited administrators to edit account information.

**Action reference**

Name	Type	Required	Description
set-content-meeting-permissions	Boolean	N	A value of <code>true</code> allows limited administrators to set the permissions for content or meetings. The default value is <code>true</code> .
compliance	Boolean	N	A value of <code>true</code> allows limited administrators to change compliance settings (settings for enabling pods, sharing, and recording, and for training settings).
chargebacks	Boolean	N	A value of <code>true</code> allows limited administrators to access the cost-center settings for this account.
view-training-reports	Boolean	N	A value of <code>true</code> allows limited administrators to view training reports.
reset-to-default	Boolean	N	A value of <code>true</code> resets all permissions to the default permissions set by Adobe.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Sample request**

The example shows

**Action reference**

```
https://admin.ibreeze.macromedia.com/api/xml?action=limited-administrator-permissions-update
&session=breezghd9nxdhh768vpob
&view-user-data=true
&view-user-data=false
&reset-password=true
&reset-password=false
&modify-current-users-groups=true
&modify-current-users-groups=false
&add-users-groups-webui=true
&add-users-groups-webui=false
&add-users-groups-csv=false
&delete-users-groups=false
&user-profile-fields=true
&user-profile-fields=false
&change-login-pw-policy=true
&change-login-pw-policy=false
&chargebacks=false
&edit-account-info=true
&edit-account-info=false
&quota-threshold-notifications=false
&customization=false
&view-disk-usage-and-reports=false
&view-system-usage-reports=false
&compliance=false
&set-content-meeting-permissions=true
&set-content-meeting-permissions=false
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8"?>
<results><status code="ok"/>
</results>
```

## login

**Availability**

Breeze 4

**Description**

Logs a user in to Adobe Connect Server.

In a client application, after logging in a user, you must read and store the cookie called `BREEZESSESSION`, which can be found in the HTTP headers of the response from `login`. You must then include the value of that cookie in every subsequent request that you make for that user.

If you cannot retrieve cookie values from HTTP response headers, you can call `common-info` to get the cookie value before the user logs in. Then, pass the value to `login` using the `session` request parameter:

```
https://example.com/api/xml?action=login&login=loginId&password=password
&session=value
```

You can also use the `session` parameter on any API call you make after `login`. For example, to call `principal-list` after logging in, you can enter:

```
https://example.com/api/xml?action=principal-list&session=value
```

**Action reference**

The `BREEZESSESSION` value is valid for only one login session. Your application must store a new cookie value each time the user logs in.

When you call the `login` action, you are sending a login ID and password across a network, unless you use external authentication. Use SSL or another appropriate security method to protect passwords in transit.

**Request URL**

```
http://server_name/api/xml
    ?action=login
    &login=string
    &password=string
    &account-id=integer
    &external-auth=use
    &domain=string
```

**Parameters**

Name	Type	Required	Description
account-id	Integer	N	The ID of your Adobe Connect hosted account. If your organization is running a licensed Adobe Connect Server, do not use <code>account-id</code> .
external-auth	Allowed value	N	A value indicating whether you send an external network login ID to represent the user to Adobe Connect. If so, use <code>external-auth=use</code> .
login	String	Y/N	The user's login name. Do not use if you use external or HTTP header authentication.
password	String	Y/N	The user's password. Do not use if you use external or HTTP header authentication.
domain	String	N	The domain name of your Adobe Connect hosted account. If your organization is running a licensed of Adobe Connect Server, do not use <code>domain</code> .
session	String	N	The value of the <code>BREEZESSESSION</code> cookie . Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
    <status code=allowedValue />
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<code>status</code>		Empty, with attributes	The status of the response.
	<code>code</code>	Allowed value	A code indicating the response status (see <code>status</code> ).

**Action reference**

**Sample request**

```
http://example.com/api/xml?action=login&login=joy@acme.com&password=happy
    &session=breeztg8mz53r93vebwur
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
    <status code="ok" />
</results>
```

**See also**

[logout](#)

## logout

**Availability**

Breeze 4

**Description**

Ends a user’s login session, invalidating the cookie value associated with the user’s session.

After calling `logout`, set the `BREEZESESSION` cookie value to `null`. Do not reuse the cookie value after your user logs out.

**Request URL**

```
http://server_name/api/xml
    ?action=logout
    &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
session	String	N	The value of the <code>BREEZESESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
    <status code=allowedValue />
</results>
```



**Action reference****Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Sample request**

```
http://example.com/api/xml?action=logout
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
</results>
```

**See also**

[login](#)

## meeting-disclaimer-info

**Availability**

Acrobat Connect Pro 7

**Description**

Provides information about the disclaimer text that is shown when a user enters a meeting. For more information about the disclaimer, see [meeting-disclaimer-update](#).

**Request URL**

```
https://servername/api/xml
  ?action=meeting-disclaimer-info
  &account-id=integer
  &session=string
```

**Parameters**

Name	Type	Required	Description
account-id	Integer	N	The ID of the account for which the disclaimer text is retrieved. If not used, the account that you are currently logged in to is updated.
session	String	N	A string; the value of the BREEZESSION cookie.

**Filters**

Filters cannot be used with this action.

**Action reference****Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <disclaimer>
    string
  </disclaimer>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
disclaimer		String	The text of the disclaimer notice.

**Sample request**

```
https://example.com/api/xml?action=meeting-disclaimer-info&account-id=7
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <disclaimer>
    This meeting may be recorded for compliance purpose. By clicking OK you agree to
the terms of meeting.
  </disclaimer>
</results>
```

**meeting-disclaimer-update****Availability**

Acrobat Connect Pro 7

**Description**

Updates the disclaimer text that is shown when a user enters a meeting.

To comply with communications regulations or standards, you can set up a disclaimer notice to appear when a user enters a meeting. The disclaimer notice typically displays boilerplate information for your organization. It advises users of the status of the meeting and the terms of use for the meeting. For example, a disclaimer notice could advise users that the meeting is being recorded, and that users cannot join the meeting unless they accept the notice.

If the disclaimer is activated, the notice is shown in all meetings. Activate the disclaimer either through the management console or by using the [meeting-feature-update](#) action with the `fid-meeting-disclaimer` parameter set to `enabled`.

**Action reference**

**Request URL**

```
https://servername/api/xml
    ?action=meeting-disclaimer-update
    &account-id=integer
    &disclaimer=string
    &session=string
```

**Parameters**

Value	Type	Required	Description
account-id	Integer	N	The ID of the account for which the disclaimer text is updated. If not used, the account that you are currently logged into is updated.
disclaimer	String	Y	The disclaimer text that is shown when a user starts a meeting. The disclaimer can, for example, notify users that a meeting is being recorded.  The limit is 1500 characters. The disclaimer text can contain XML-compliant HTML tags. For example: <b>This meeting is being recorded.</b>
session	String	N	The value of the BREEZESSESSION cookie.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
    <status code=code />
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Sample request**

```
https://example.com/api/xml?action=meeting-disclaimer-update&disclaimer=Please note that this meeting is being recorded.
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
    <results>
        <status code="ok" />
    </results>
```

**meeting-feature-update**

**Availability**

Acrobat Connect Pro 7

**Action reference**

**Description**

Enables or disables features in a meeting. This action is used to manage features such as recording of meetings and control of pods. For more information on usage, see “[Configure compliance settings](#)” on page 44. You can append multiple `feature-id` and `enable` pairs to the end of the request URL.

**Request URL**

```
http://server name/api/xml
    ?action=meeting-feature-update
    &account-id=integer
    &feature-id=value
    &enable=value
```

**Parameters**

Name	Type	Required	Description
account-id	Integer	Y	The ID of your Adobe Connect hosted account. For enterprise installations, the ID is 7. For licensed installations, use <a href="#">common-info</a> to get the ID.
<a href="#">feature-id</a>	Integer	Y	The ID of the feature to enable or disable. For available IDs, see <a href="#">feature-id</a> .
enable	Boolean	Y	Whether to enable the specified feature (true) or not (false).

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
    <status code=code />
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Sample request**

The following sample disables the Chat pod.

```
https://example.com/api/xml?action=meeting-feature-update&account-id=7&feature-id=fid-meeting-chat&enable=false
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
    <results>
        <status code="ok" />
    </results>
```

## permissions-info

### Availability

Breeze 4

### Description

Returns the list of principals (users or groups) who have permissions to act on a SCO, principal, or account.

To call `permissions-info`, you must specify an `acl-id`, which is the ID of a SCO, principal, or account that can be acted on. ACL stands for *access control list*, and means the list of entities who have permission.

With just an `acl-id`, `permissions-info` returns a list of all principals in the account, showing each principal's permission on the principal or SCO specified in the `acl-id`:

```
https://example.com/api/xml?action=permissions-info&acl-id=2006258745
```

To check the permissions a specific principal has on a principal or SCO within an account, call `permissions-info` with an `acl-id` and a filter on `principal-id`:

```
http://example.com/api/xml?action=permissions-info&acl-id=7&filter-principal-id=10022
```

To check the permissions a principal has on an account, call `permissions-info` with both an `acl-id` (specifying an `account-id`) and a `principal-id`:

```
https://example.com/api/xml?action=permissions-info&acl-id=624520&principal-id=624523
```

### Request URL

```
http://server_name/api/xml
  ?action=permissions-info
  &acl-id=integer
  &principal-id=integer
  &filter-definition=value
  &sort-definition=value
  &session=BreezeSessionCookieValue
```

### Parameters

Name	Type	Required	Description
<code>acl-id</code>	Integer	Y	The ID of a SCO, account, or principal that a principal has permission to act on. The <code>acl-id</code> is a <code>sco-id</code> , <code>principal-id</code> , or <code>account-id</code> in other calls.
<code>principal-id</code>	Integer	N	The ID of a principal who has a permission (even if denied) to act on an object.
<code>filter-definition</code>	Filter definition	N	A filter to reduce the volume of the response.
<code>sort-definition</code>	Sort definition	N	A sort to return results in a certain sequence.
<code>session</code>	String	N	The value of the <code>BREEZESSESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

### Filters

You can filter or sort the response on any element or attribute it contains.

**Action reference**

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <permissions>
    <principal principal-id=integer is-primary=boolean type=allowedValue
      has-children=boolean permission-id=integer training-group-id=integer>
      <name>string</name>
      <login>string</login>
    </principal></permissions>
    ...
  <permission acl-id=integer permission-id=allowedValue
    principal-id=integer />
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
permissions		Container	A list of principals showing their permission to access the SCO, account, or principal.
principal		Container	Information about one principal showing the principal's permission level on the SCO, account, or principal.
	principal-id	Integer	The ID of a principal who has permission on a SCO, account, or principal.
	is-primary	Boolean	A value indicating whether the principal is a primary group (same as a built-in group).
	type	Allowed value	The type of principal (see <a href="#">type</a> for allowed values).
	has-children	Boolean	A value indicating whether the principal has children. Groups have children and users don't, so if <code>true</code> , the principal is a group.
	permission-id	Allowed value	The permission the principal has on the SCO, account, or principal (see <a href="#">permission-id</a> for values).
	acl-id	Integer	The ID of the SCO on which the permission is defined.
name		String	The name of the principal who has permission to access the SCO.
login		String	The login name of the principal who has permission to access the SCO.
permission		Empty, with attributes	Information about the permission one principal has on a SCO, account, or principal. If empty, no permission is defined.
	acl-id	Integer	The ID of the object on which the principal has permission.
	permission-id	Allowed value	The permission the principal has to act on the object (see <a href="#">permission-id</a> for values).
	principal-id	Integer	The ID of the principal who has permission to act on the object.
	training-group-id	Integer	The ID of the training group.

**Action reference**

**Sample request**

https://example.com/api/xml?action=permissions-info&acl-id=2006334033

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <permissions>
    <principal principal-id="2006258745" is-primary="false" type="user"
      has-children="false" permission-id="host" training-group-id="2007842424">
      <name>Joy Smith</name>
      <login>joy@acme.com</login>
    </principal>
    ...
  </permissions>
</results>
```

**See also**

[permissions-reset](#), [permissions-update](#)

## permissions-reset

**Availability**

Breeze 4

**Description**

Resets all permissions any principals have on a SCO to the permissions of its parent SCO. If the parent has no permissions set, the child SCO will also have no permissions.

**Request URL**

```
http://server_name/api/xml
  ?action=permissions-reset
  &acl-id=integer
  &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
acl-id	Integer	Y	The ID of a SCO that has permissions you want to reset.
session	String	N	The value of the BREEZESESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=code />
</results>
```

**Action reference****Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Sample request**

```
https://example.com/api/xml?action=permissions-reset&acl-id=2006334033
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
</results>
```

**See also**

[permissions-info](#), [permissions-update](#)

## permissions-update

**Availability**

Breeze 4

**Description**

Updates the permissions a principal has to access a SCO, using a trio of `principal-id`, `acl-id`, and `permission-id`. To update permissions for multiple principals or objects, specify multiple trios. You can update more than 200 permissions in a single call to `permissions-update`.

Call `permissions-update` to give a user access to a Adobe Connect meeting, course, curriculum, or other SCO. For example, you can use `permissions-update` to:

- Invite a user to a meeting as participant, presenter, or host (with a `permission-id` of `view`, `mini-host`, or `host`, respectively)
- Remove a user's participant, presenter, or host access to a meeting (with a `permission-id` of `remove`)
- Enroll users in courses (with a `permission-id` of `view`)

If you use multiple trios and any of them have invalid information (for example, an incorrect `acl-id` or `principal-id`), `permissions-update` returns an `ok` status, the correct trios execute, and the invalid ones do not.

**Request URL**

```
http://server_name/api/xml
  ?action=permissions-update
  &acl-id=integer
  &principal-id=integer
  &permission-id=allowedValue
  &session=BreezeSessionCookieValue
```



**Action reference**

**Parameters**

Name	Type	Required	Description
acl-id	Integer	Y	The ID of a SCO (a <code>sco-id</code> ) for which you want to update permissions.
principal-id	Integer	Y	The ID of a principal, either a user or group.
permission-id	String	Y	The permission to assign (see <a href="#">permission-id</a> for values).
session	String	N	The value of the <code>BREEZESESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
<status code=code />
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Sample request**

```
https://example.com/api/xml?action=permissions-update&acl-id=2006334033
&principal-id=2006258745&permission-id=host
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
</results>
```

**See also**

[permissions-info](#), [permissions-reset](#)

**principal-info**

**Availability**

Breeze 4

**Description**

Provides information about one principal, either a user or a group.

**Action reference**

You must specify a `principal-id`. To find the `principal-id`, call `principal-list`, using a filter if necessary to limit the response.

**Request URL**

```
http://server_name/api/xml
?action=principal-info
&principal-id=integer
&session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
<code>principal-id</code>	Integer	Y	The ID of a user or group you want information about. You can get the ID by calling <code>principal-list</code> .
<code>session</code>	String	N	The value of the <code>BREEZESSESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Action reference**

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <contact>
    <email>string</email>
    <first-name>string</first-name>
    <last-name>string</last-name>
  </contact>
  <manager account-id=integer disabled=boolean has-children=boolean
    is-hidden=boolean is-primary=boolean principal-id=integer
    type=allowedValue>
    <ext-login>string</ext-login>
    <login>string</login>
    <name>string</name>
    <email>string</email>
    <first-name>string</first-name>
    <last-name>string</last-name>
    <x-customfield1>string</x-customfield1>
    <x-customfield2>string</x-customfield2>
    ...
  </manager>
  <preferences acl-id=integer lang=allowedValue
    time-zone-id=allowedValue />
  <principal account-id=integer disabled=boolean has-children=boolean
    is-hidden=boolean is-primary=boolean principal-id=integer
    type=allowedValue>
    <description>string</description>
    <ext-login>string</ext-login>
    <login>string</login>
    <name>string</name>
    <email>string</email>
    <first-name>string</first-name>
    <last-name>string</last-name>
    <x-customfield1>string</x-customfield1>
    <x-customfield2>string</x-customfield2>
    ...
  </principal>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	Top-level element for the response.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
contact		Container	Information about the contact person for a principal. If the principal is a user, usually the same as information in <a href="#">principal</a> .
email		String	The e-mail address of the contact person.
first-name		String	The first name of the contact person.
last-name		String	The last name of the contact person.

Action reference

Element	Attribute	Type	Description
manager		Container	Information describing a user's manager, who is also a principal.
principal		Container	Information describing the principal.
	account-id	Integer	The ID of the account the principal belongs to.
	disabled	Datetime	If the principal's account is valid, a null value returned as "". If the account is disabled, the date it was disabled.
	has-children	Boolean	Whether the principal has children. Groups have children and users don't, so this attribute indicates whether the principal is a group.
	is-hidden	Boolean	Whether the principal is hidden ( <code>true</code> ) or not ( <code>false</code> ) in Connect Central or your application.
	is-primary	Boolean	Whether the principal is a built-in group ( <code>true</code> ) or not ( <code>false</code> ).
	principal-id	Integer	The ID of the principal.
	type	Allowed value	The type of principal (see <a href="#">type</a> for values).
description		String	For a group, the group name.
ext-login		String	For a user, the login ID sent from an external network. By default, the same value as <code>login</code> , so change it if you use external authentication.
login		String	The principal's login ID on Adobe Connect. Can be the same as an e-mail address.
name		String	For a user, the full name, concatenated from <code>first-name</code> and <code>last-name</code> .
email		String	For a user, the e-mail address.
first-name		String	For a user, the first name.
last-name		String	For a user, the last name.
x-customfield		String	A custom field defined for the user or group.
preferences		Empty, with attributes	Information about the principal's preferences.
	acl-id	Integer	The principal's ID.
	lang	Allowed value	The language setting the principal has chosen for Adobe Connect applications.
	time-zone-id	Allowed value	The time zone setting the principal has chosen for Adobe Connect applications.

Sample request

<https://example.com/api/xml?action=principal-info&principal-id=2006258745>

**Action reference****Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <contact>
    <email>bob@acme.com</email>
    <first-name>Bob</first-name>
    <last-name>Jones</last-name>
  </contact>
  <manager account-id="624520" disabled="" has-children="false"
    is-hidden="false" is-primary="false" principal-id="2006282569"
    type="user">
    <ext-login>jazzdoe@example.com</ext-login>
    <login>jazzdoe@example.com</login>
    <name>jazz doe</name>
    <email>jazzdoe@example.com</email>
    <first-name>Jazz</first-name>
    <last-name>Doe</last-name>
    <x-2007017651>San Francisco</x-2007017651>
  </manager>
  <preferences acl-id="2006258745" lang="en" time-zone-id="4" />
  <principal account-id="624520" disabled="" has-children="false"
    is-hidden="false" is-primary="false" principal-id="2006258745"
    type="user">
    <ext-login>joy@acme.com</ext-login>
    <login>joy@acme.com</login>
    <name>Joy Smith</name>
    <email>joy@acme.com</email>
    <first-name>Joy</first-name>
    <last-name>Smith</last-name>
    <x-2007017651>San Francisco</x-2007017651>
  </principal>
</results>
```

**See also**

[principal-list](#), [principal-list-by-field](#), [principal-update](#)

**principal-list****Availability**

Breeze 4

**Description**

Provides a complete list of users and groups, including primary groups.

This call is useful for getting a `principal-id` when you don't have one. However, be aware that it returns a list of all principals on your Adobe Connect Server or Adobe Connect hosted account, unless you use a filter to limit the response.

You can also use `principal-list` to get a list of groups in an account by filtering on the `type` and `is-member` fields:

```
https://example.com/api/xml?action=principal-list&filter-type=group
  &filter-is-member=true
```

**Action reference**

However, `filter-type=group` returns groups you have created, not built-in groups predefined on the server. Built-in groups have `type` values other than `group`, such as `admins` and `authors` (see [type](#) for a list of the values).

You can filter the response with a `filter-type` parameter set to the type of group you want, then parse the response for a `principal-id`, then pass the `principal-id` as a `group-id` on another request to `principal-list`.

**Request URL**

```
http://server_name/api/xml
  ?action=principal-list
  &group-id=integer
  &filter-definition=value
  &sort-definition=value
  &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
<code>group-id</code>	Integer	N	The ID of a group. Same as the <code>principal-id</code> of a principal that has a <code>type</code> value of <code>group</code> .
<a href="#">filter-definition</a>	Filter definition	N	A filter to reduce the volume of the response.
<a href="#">sort-definition</a>	Sort definition	N	A sort to return results in a certain sequence.
<code>session</code>	String	N	The value of the <code>BREEZESSESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

*Note: Filtering on the `login` element is useful but slow, and reduced performance is unavoidable.*

You can also filter on a special field name, `manager-id`, to return a list of principals who report to a given manager, for example:

```
https://example.com/api/xml?action=principal-list
  &filter-manager-id=2006282569
```

When you use `filter-manager-id`, each `principal` element in the response has a `manager-id` attribute:

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <principal-list>
    <principal principal-id="2006258745" account-id="624520" type="user"
      has-children="false" is-primary="false" is-hidden="false"
      manager-id="2006282569">
      <name>Pat Lee</name>
      <login>plee@mycompany.com</login>
      <email>plee@mycompany.com</email>
    </principal>
  </principal-list>
</results>
```

**Action reference**

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <principal-list>
    <principal principal-id=integer account-id=integer type=allowedValue
      has-children=boolean is-primary=boolean is-hidden=boolean
      manager-id=integer training-group-id=integer>
      <name>string</name>
      <login>string</login>
      <email>string</email>
    </principal>
  </principal-list>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
principal-list		Container	The entire list of principals.
principal		Container	Details about one principal.
	principal-id	Integer	The ID of the principal.
	account-id	Integer	The ID of the account the principal belongs to.
	type	Allowed value	The type of principal (see <a href="#">type</a> for values).
	has-children	Boolean	Indicates whether the principal has children. Groups have children and users do not, so when <code>has-children</code> is <code>true</code> , the principal is a group.
	is-primary	Boolean	Whether the principal is a built-in group ( <code>true</code> ) or not ( <code>false</code> ).
	is-hidden	Boolean	Whether Connect Central or your application displays the principal ( <code>true</code> for not displayed and <code>false</code> for displayed).
	manager-id	Integer	The <code>principal-id</code> of the manager the principal reports to. Returned only if you use <code>filter-manager-id</code> in the request.
	training-group-id=	Integer	The ID of the training group.
name		String	The principal's full name.
login		String	The principal's login ID, often an e-mail address.
email		String	The principal's e-mail address.
principal-custom-field-values		Container	The entire list of custom field values defined for the principal.
field		Container	Details about one custom field defined for the principal (see <a href="#">field</a> for contents).

**Action reference****Sample request**

```
https://example.com/api/xml?action=principal-list
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <principal-list>
    <principal principal-id="624526" account-id="624520" type="user"
      has-children="false" is-primary="false" is-hidden="false" training-group-
id="">
      <name>ned mack</name>
      <login>nmack@acme.com</login>
      <email>nmack@acme.com</email>
    </principal>
    <principal principal-id="624550" account-id="624520" type="user"
      has-children="false" is-primary="false" is-hidden="false" training-group-
id="">
      <name>amelie jones</name>
      <login>amelie@example.com</login>
      <email>amelie@example.com</email>
    </principal>
    ...
  </principal-list>
</results>
```

**See also**

[principal-info](#), [principal-update](#), [principal-list-by-field](#)

**principal-list-by-field****Availability**

Breeze 5

**Description**

Lists principals that have a specified value in a custom field. Use this action to query custom fields for principals. Use `principal-list` to get a list of custom fields that are defined for the principal.

In the `value` parameter, enter the value of a custom database field. The `name` element returned by `principal-list`, for example, is a full name concatenated from the `first-name` (*bob*) and `last-name` (*jones*) database fields. If you search on *bob jones*, `principal-list-by-field` does not return a value, unless the full name is defined as a database field (in this case, a custom field defined on principals).

The search is case insensitive, and the query string can contain spaces.

Wildcards are not allowed in the query string. For example, if you enter `t*`, `principal-list-by-field` searches for the exact string `t*`.

The `principal-list-by-field` action searches in all custom database fields defined for the principal; it does not search principal fields.



**Action reference**

**Request URL**

```
http://server_name/api/xml
    ?action=principal-list-by-field
    &value=string
    &filter-definition=value
    &sort-definition=value
    &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
value	String	Y	The value for which you want to search all fields. You do not need to enter a field name.
<a href="#">filter-definition</a>	Filter definition	N	A filter to reduce the volume of the response.
<a href="#">sort-definition</a>	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <principal-list>
    <principal account-id=integer principal-id=integer type=allowedValue
      has-children=boolean is-primary=boolean is-hidden=boolean>
      <name>string</name>
      <login>string</login>
    </principal>
  </principal-list>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
principal-list		Container	The entire list of principals that match the value in one or more custom fields.
principal		Container	One principal that matches the value.
	principal-id	Integer	The ID of the principal.
	account-id	Integer	The ID of the account the principal belongs to.
	type	Allowed value	The type of principal (see <a href="#">type</a> for values).

**Action reference**

Element	Attribute	Type	Description
	has-children	Boolean	Indicates whether the principal has children. Groups have children and users don't, so this attribute indicates whether the principal is a group.
	is-primary	Boolean	Whether the principal is a built-in group ( <code>true</code> ) or not ( <code>false</code> ).
	is-hidden	Boolean	Whether the principal is hidden in the user interface ( <code>true</code> ) or not ( <code>false</code> ).
name		String	The principal's full name, concatenated from the <code>first-name</code> and <code>last-name</code> fields.
login		String	The principal's login ID, often an e-mail address.

**Sample request**

```
https://example.com/api/xml?action=principal-list-by-field&value=inactive
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <principal-list>
    <principal account-id="624520" principal-id="2616099" type="user"
      has-children="false" is-primary="false" is-hidden="false">
      <name>Bob Jones</name>
      <login>bjones@acme.com</login>
    </principal>
  </principal-list>
</results>
```

**See also**

[principal-info](#), [principal-list](#), [principal-update](#)

## principals-delete

**Availability**

Breeze 4

**Description**

Removes one or more principals, either users or groups. To delete principals, you must have Administrator privilege.

To delete multiple principals, specify multiple `principal-id` parameters. All of the principals you specify will be deleted.

The `principal-id` can identify either a user or group. If you specify a user, the user is removed from any groups the user belongs to. If you specify a group, the group is deleted, but the users who belong to it are not.

**Request URL**

```
http://server_name/api/xml
?action=principals-delete
&principal-id=integer
&session=BreezeSessionCookieValue
```

**Action reference****Parameters**

Name	Type	Required	Description
principal-id	Integer	Y	The ID of a user or group you want to delete.
session	String	N	The value of the BREEZESESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Sample request**

```
https://example.com/api/xml?action=principals-delete
&principal-id=2006339311&principal-id=2006339323
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
</results>
```

**See also**

[principal-info](#), [principal-list](#), [principal-list-by-field](#), [principal-update](#)

## principal-update

**Availability**

Breeze 4

**Description**

Creates a principal (a user or group) or updates a standard field for a principal. The principal is created or updated in the same account as the user making the call.

To create a new principal, call `principal-update` without specifying a `principal-id`. To update, add the `principal-id`. Before you update metadata about a principal, call [principal-info](#) to get the existing version.

If a principal has custom fields, use [acl-field-update](#) to update them, rather than `principal-update`.

**Action reference**

You need Administrator privileges to create or update a principal.

**Request URL**

```
http://server_name/api/xml
  ?action=principal-update
  &description=string
  &email=string
  &first-name=string
  &has-children=boolean
  &last-name=string
  &login=string
  &name=string
  &password=string
  &principal-id=integer
  &send-email=boolean
  &type=allowedValue
  &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
description	String	N	The new group's description. Use only when creating a new group.
email	String	N	The user's e-mail address. Can be different from the login. Be sure to specify a value if you use send-email=true.
first-name	String	Y/N	The user's new first name. Use only with users, not with groups. Required to create a user.
has-children	Boolean	Y	Whether the principal has children. If the principal is a group, use 1 or true. If the principal is a user, use 0 or false.
last-name	String	Y/N	The new last name to assign to the user. Required to create a user. Do not use with groups.
login	String	Y/N	The principal's new login name, usually the principal's e-mail address. Must be unique on the server. Required to create or update a user. Do not use with groups.
name	String	Y/N	The new group's name. Use only when creating a new group. Required to create a group.
password	String	N	The new user's password. Use only when creating a new user.
principal-id	String	Y/N	The ID of the principal that has information you want to update. Required to update a user or group, but do not use to create either.
send-email	Boolean	N	A flag indicating whether the server should send an e-mail to the principal with account and login information.
type	String	Y/N	The type of principal. Use only when creating a new principal (see <a href="#">type</a> for values).
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Action reference**

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <principal principal-id=integer account-id=integer
    has-children=integer type=integer>
    <login>string</login>
    <ext-login>string</ext-login>
    <name>string</name>
  </principal>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
principal		Container	Information about the newly created principal.
	principal-id	Integer	The ID of the newly created user.
	account-id	Integer	The ID of the account the new user belongs to. Same as the account of the current user.
	has-children	Boolean	Whether the principal has children, which indicates whether the principal is a user or group (1 if a group, or 0 if a user).
	type	Allowed value	The type of principal (see <a href="#">type</a> for values).
login		String	The principal's login ID, often an e-mail address.
ext-login		String	The principal's external authentication ID. By default, the value is the same as login, unless you explicitly set the value to an authentication ID from your network.
name		String	The principal's name. If the principal is a user, concatenated from the <code>first-name</code> and <code>last-name</code> parameters in the request.

**Sample request**

```
https://example.com/api/xml?action=principal-update&first-name=jake
  &last-name=doe&has-children=0&login=jakedoe@example.com&type=user
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <principal principal-id="2006403978" account-id="624520" type="user"
    has-children="0">
    <login>jakedoe@example.com</login>
    <ext-login>jakedoe@example.com</ext-login>
    <name>jake doe</name>
  </principal>
</results>
```

**See also**

[principal-info](#), [principal-list](#), [principal-list-by-field](#), [acl-field-update](#)

**quota-threshold-info****Availability**

Acrobat Connect Pro 7

**Description**

Provides the list of quotas for which capacity notifications are provided, along with their current threshold settings.

Each Adobe Connect account has system quotas that determine, for example, how many seats are available for Meeting Hosts, Learners, and so on. Each quota has a threshold; when the threshold is crossed, the system notifies administrators that the quota is in danger of being reached. The settings for the threshold and the notifications vary depending on the quota.

**Request URL**

```
https://example.com/api/xml
  ?action=quota-threshold-info
  &account-id=integer
  &session=integer
```

**Parameters**

Name	Type	Required	Description
account-id	Integer	N	The ID of the account for which you want quota threshold information. If you don't specify an ID, the current account to which the user is logged in is used.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <Principals>
    <Principal principal-id="integer" type="string"/>
  </Principals>
  <Quotas>
    <Quota acl-id="integer" quota-id="string" threshold-pct="integer" login-
notif="boolean" email-notif="boolean" monthly-emails="boolean" limit="integer"
used="integer"/>
    <Quota acl-id="integer" quota-id="string" threshold-pct="integer" login-
notif="boolean" email-notif="boolean" monthly-emails="boolean" limit="integer"
used="integer"/>
  </Quotas>
  <Trees>
    <Tree tree-id="integer" type="string"/>
  </Trees>
</results>
```

Action reference

Response values

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
Principals		Container	Lists the principals specifying the groups for which system capacity notifications are provided.
Principal		Container	Information about the principal for which system capacity notification is provided.
	principal-id	Integer	The principal ID specifying the group.
	type	String	The group type. Depending on the license, this value can be one of the following: <ul style="list-style-type: none"> <li>• authors</li> <li>• live-admins, which specifies that the group is Meeting Hosts</li> </ul>
Quotas		Container	Lists the quotas.
Quota		Container	Information about the quota and its settings.
	acl-id	Integer	ACL ID of the quota.
	quota-id	Integer	The ID of the quota. For possible values, see <a href="#">quota-ID</a> .
	threshold-pct	Integer	The percent value of the threshold.
	login-notif	Boolean	Whether administrators are notified upon logging in that a threshold is exceeded ( <code>true</code> ) or not ( <code>false</code> ).
	email-notif	Boolean	Whether administrators are notified through e-mail that a threshold is exceeded ( <code>true</code> ) or not ( <code>false</code> ).
	monthly-emails	Boolean	Whether administrators are sent monthly threshold reports through e-mail ( <code>true</code> ) or not ( <code>false</code> ).
	limit	Integer	The limit of member seats.
	used	Integer	Number of member seats used.
Trees		Container	Provides information about the tree type quotas (the quota for the number of concurrent users per meeting).
Tree		Container	Information about the tree.
	tree-id	Integer	The tree ID.
	type	String	The tree type, which is one of the following values: <ul style="list-style-type: none"> <li>• meetings</li> <li>• user-meetings</li> </ul>

Sample request

<https://example.com/api/xml?action=quota-threshold-info&account-id=7>

**Action reference****Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok"/>
  <Principals>
    <Principal principal-id="20013" type="authors"/>
    <Principal principal-id="10051" type="live-admins"/>
  </Principals>
  <Quotas>
    <Quota acl-id="7" quota-id="training-user" threshold-pct="90" login-notif="true"
email-notif="true" monthly-emails="true" limit="10" used="0"/>
    <Quota acl-id="20013" quota-id="num-of-members-quota" threshold-pct="10" login-
notif="true" email-notif="true" monthly-emails="true" limit="10" used="3"/>
  </Quotas>
  <Trees/>
</results>
```

**quota-threshold-exceeded****Availability**

Acrobat Connect Pro 7

**Description**

Returns information about system quota thresholds that have been exceeded.

Each Adobe Connect account has system quotas that determine, for example, how many seats are available for Meeting Hosts, Learners, and so on. Each quota has a threshold; when the threshold is crossed, the system notifies administrators that the quota is in danger of being reached. The threshold varies depending on the quota. For more information about automatic notification, see *Adobe Connect User Guide*.

**Request URL**

```
https://example.com/api/xml
?action=quota-threshold-exceeded
&account-id=Integer
&acl-id=Integer
&quota-id=String
&num-of-days=Integer
&session=String
```



**Action reference**

**Parameters**

Name	Type	Required	Description
account-id	Integer	N	The ID of the account. Specify either account-id or acl-id (not both). If you do not specify either account-id or acl-id, results are returned for the account which the current user is logged into.  If you specify account-ID, results are returned for all quota thresholds that have been reached in the account.
acl-id	Integer	N	The ID of the SCO, account, or principal for which you want threshold information. Can be a valid sco-id, account-id, or principal-id. If you do not specify a value for acl-id, the value for account-id is used.  The value to use for acl-id depends on the quota ID used; for more information, see "quota-ID" on page 240
quota-id	String	N	The ID of the system quota for which you want information. For available values, see "quota-ID" on page 240.
num-of-days	Integer	N	Number of days from the current day for which records are retrieved. If you do not specify a value, all the previous records for the specified quotas are retrieved.
session	String	N	The value of the BREZESESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <Records>
    <Record acl-id="Integer" quota-id="String" peak-used="Integer" count="Integer"
threshold-pct="Integer" sco-id="Integer">
      <record-date>Date</record-date>
    </Record>
  </Records>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see status).
Records		Container	Lists the records returned.
Record		Container	Lists information about the record returned.
	acl-id	Integer	The ID of the ACL returned.
	quota-id	String	The ID of the quota returned. For available values, see quota-ID.
	peak-used	Integer	The peak value of quota used on the specified date. This attribute is null for training and group quotas.
	count	Integer	Number of times the threshold was crossed for the quota on the specified date. This attribute is null for training and group quotas.

**Action reference**

Element	Attribute	Type	Description
	threshold-pct	Integer	Percentage threshold when the threshold was crossed.
	sco-id	Integer	The ID of the meeting for which the threshold was crossed. This attribute is applicable only for the quota ID <code>concurrent-users-per-meeting</code> ; for other quotas, it is null.
Record-date		Date	Date when the threshold was crossed (UTC), in MM/DD/YYYY format.

**Sample request**

```
https://example.com/api/xml?action=quota-threshold-exceeded&acl-id=20013&quota-id=num-of-members-quota&num-of-days=30
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
<status code="ok"/>
  <Records>
    <Record acl-id="20013" quota-id="num-of-members-quota" peak-used="" count=""
threshold-pct="10" sco-id="">
      <record-date>11/20/2007</record-date>
    </Record>
  </Records>
</results>
```

**quota-threshold-update****Availability**

Acrobat Connect Pro 7

**Description**

Updates the threshold settings of the specified quotas.

Each Adobe Connect account has system quotas that determine, for example, how many seats are available for meeting hosts, trainers, training managers, and so on. Each quota has a threshold; when the threshold is crossed, the system notifies administrators that the quota is in danger of being reached. The settings for the thresholds and notifications vary depending on the quota, and you can configure the settings using this action.

**Request URL**

```
https://example.com/api/xml
?action=quota-threshold-update
&account-id=integer
&acl-id=integer
&quota-id=string
&threshold-pct=integer
&login-notif=Boolean
&email-notif=Boolean
&monthly-emails=Boolean
```

**Action reference**

**Parameters**

Name	Type	Required	Description
account-id	Integer	N	The ID of the account for which quota settings are updated.
acl-id	Integer	Y	The ID of the SCO, account, or principal for which you want threshold information. Can be a valid <code>sco-id</code> , <code>account-id</code> , or <code>principal-id</code> . If you do not specify a value for <code>acl-id</code> , the value for <code>account-id</code> is used.  The value to use for <code>acl-id</code> depends on the quota ID used; for more information, see <a href="#">quota-ID</a> .
quota-id	String	Y	The ID of the quota whose settings are updated. For available values, see <a href="#">quota-ID</a> .
threshold-pct	Integer	Y	The percent threshold for the quota. The lower the value, the more frequently administrators are notified when the threshold is exceeded (if notifications are enabled).
login-notif	Boolean	Y	Specifies whether to notify administrators upon logging in that a threshold is exceeded ( <code>true</code> ) or not ( <code>false</code> ).
email-notif	Boolean	Y	Specifies whether to notify administrators through e-mail that a threshold is exceeded ( <code>true</code> ) or not ( <code>false</code> ).
monthly-emails	Boolean	Y	Specifies whether to send administrators monthly threshold reports through e-mail ( <code>true</code> ) or not ( <code>false</code> ).
session	String	N	The value of the <code>BREEZESSESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Sample request**

```
https://example.com/api/xml?action=quota-threshold-update&account-id=7&acl-id=7&quota-id=training-user&threshold-pct=90&login-notif=false&email-notif=true&monthly-emails=true
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
</results>
```

**See also**

[quota-threshold-info](#), [quota-threshold-exceeded](#)

## report-active-meetings

### Availability

Breeze 4

### Description

Returns a list of Adobe® Connect™ meetings that are currently in progress, including the number of minutes the meeting has been active.

For `report-active-meetings` to return results, at least one user must be present in at least one meeting room. If meetings are scheduled at present, but no users are attending those meetings, `report-active-meetings` returns an empty response.

### Request URL

```
http://server_name/api/xml
?action=report-active-meetings
&session=BreezeSessionCookieValue
```

### Parameters

Name	Type	Required	Description
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

### Filters

Results cannot be filtered or sorted.

### Response structure

```
<results>
  <status code=allowedValue />
  <report-active-meetings>
    <sco sco-id=integer active-participants=integer
      length-minutes=integer>
      <name>string</name>
      <url-path>string</url-path>
      <date-begi>datetime</date-begi>
    </sco>
  </report-active-meetings>
</results>
```

### Response values

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-active-meetings		Container	The list of all meetings currently in progress.
sco		Container	Information about one meeting in progress.

**Action reference**

Element	Attribute	Type	Description
	sco-id	Integer	The unique ID of a meeting in progress.
	active-participants	Integer	The number of users attending the meeting in progress, including hosts and presenters.
	length-minutes	Integer	The number of minutes the meeting has been active.
name		String	The name of the meeting, defined when the meeting was created.
url-path		String	The part of the meeting URL that comes after the domain and is unique to this meeting.
date-begin		Datetime	The date and time the meeting began.

**Sample request**

`https://example.com/api/xml?action=report-active-meetings`

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-active-meetings>
    <sco sco-id="2006778715" active-participants="" length-minutes="1">
      <name>Designing Online Courses</name>
      <url-path>/online/</url-path>
      <date-begin>2006-06-28T14:35:21.307-07:00</date-begin>
    </sco>
  </report-active-meetings>
</results>
```

**report-bulk-consolidated-transactions****Availability**

Breeze 5

**Description**

Returns information about principal-to-SCO transactions on your Adobe Connect server or in your Adobe Connect hosted account.

A transaction is an instance of one principal visiting one SCO. The SCO can be a Adobe Connect meeting, course, document, or any content on the server.

These are all examples of transactions:

- If a principal attends a meeting twice, two transactions exist: one for each time the principal attended the meeting.
- If five people attend a meeting, five transactions exist: one for each user who attended the meeting.
- If a principal takes two courses three times each and passes each only on the third try, six transactions exist: one for each attempt on each course.

This call returns all transactions, so consider using a filter to reduce the volume of the response. For example, if you use `filter-type=meeting`, the call returns all meeting transactions:

**Action reference**

```
https://example.com/api/xml?action=report-bulk-consolidated-transactions
&filter-type=meeting
```

From the response, you can calculate Adobe Connect meeting usage by comparing times in `date-created` and `date-closed` (see “[Calculate meeting usage](#)” on page 40”). However, this call to `report-bulk-consolidated-transactions`, with `filter-type=meeting`, returns only users who logged in to the meeting as participants, not users who entered the meeting as guests.

**Request URL**

```
http://server_name/api/xml
?action=report-bulk-consolidated-transactions
&filter-definition=value
&sort-definition=value
&session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
<code>filter-definition</code>	Filter definition	N	A filter to reduce the volume of the response.
<code>sort-definition</code>	Sort definition	N	A sort to return results in a certain sequence.
<code>session</code>	String	N	The value of the <code>BREEZESSESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-bulk-consolidated-transactions>
    <row transaction-id=integer sco-id=integer type=allowedValue
      principal-id=integer score=integer>
      <name>string</name>
      <url>relativeUrl</url>
      <login>string</login>
      <user-name>string</user-name>
      <status>allowedValue</status>
      <date-created>datetime</date-created>
      <date-closed>datetime</date-closed>
    </row>
    ...
  </report-bulk-consolidated-transactions>
</results>
```

**Action reference**

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-bulk-consolidated-transactions		Container	The entire list of transactions that matches the request.
row		Container	Details of one transaction that matches the request.
	transaction-id	Integer	The ID of the transaction.
	sco-id	Integer	The unique ID of the object (SCO) the user interacted with.
	type	Allowed value	The type of the SCO (see <a href="#">type</a> for allowed values).
	principal-id	Integer	The ID of the principal involved in the transaction.
	score	Integer	If the transaction (such as a quiz) assigned a score, the actual score. Otherwise, 0.
name		String	The name assigned to the SCO involved in the transaction.
url		String	The file name portion of the URL to the SCO involved in the transaction.
login		String	The principal's login ID.
user-name		String	The full name of the user involved in the transaction (concatenated from <i>first-name</i> and <i>last-name</i> ).
status		Allowed value	The status of the transaction. Allowed values are <i>completed</i> , <i>in-progress</i> , <i>user-passed</i> , and <i>user-failed</i> .
date-created		Datetime	The date and time the principal began interacting with the SCO and the transaction was created.
date-closed		Datetime	The date and time the principal finished interacting with the SCO and the transaction was complete.

**Sample request**

https://example.com/api/xml?action=report-bulk-consolidated-transactions  
&filter-type=meeting

**Action reference****Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-bulk-consolidated-transactions>
    <row transaction-id="2006905086" sco-id="2006905049" type="meeting"
      principal-id="2006258745" score="0">
      <name>Celebrate End of June Meeting</name>
      <url>/endjune/</url>
      <login>joy@acme.com</login>
      <user-name>Joy Smith</user-name>
      <status>completed</status>
      <date-created>2006-06-30T11:10:37.003-07:00</date-created>
      <date-closed>2006-06-30T11:45:21.397-07:00</date-closed>
    </row>
    <row transaction-id="2006905795" sco-id="2006905049" type="meeting"
      principal-id="2006258745" score="0">
      <name>Celebrate End of June Meeting</name>
      <url>/endjune/</url>
      <login>joy@acme.com</login>
      <user-name>Joy Smith</user-name>
      <status>completed</status>
      <date-created>2006-06-30T17:58:29.060-07:00</date-created>
      <date-closed>2006-06-30T17:59:09.970-07:00</date-closed>
    </row>
    ...
  </report-bulk-consolidated-transactions>
</results>
```

**See also**

[report-bulk-objects](#), [report-bulk-questions](#), [report-bulk-slide-views](#), [report-bulk-users](#)

**report-bulk-objects****Availability**

Breeze 5

**Description**

Returns information about all objects (SCOs) on a licensed Adobe Connect Server or in a Adobe Connect hosted account. The object types returned include archive, attachment, authorware, captivate, course, curriculum, external-event, flv, image, meeting, presentation, and swf.

Because the response is likely to be large, use filters to limit it. For example, to return a list of all meetings on the server, filter on the `type` field:

```
http://example.com/api/xml?action=report-bulk-objects&filter-type=meeting
```

**Request URL**

```
http://server_name/api/xml
  ?action=report-bulk-objects
  &filter-definition=value
  &sort-definition=value
  &session=BreezeSessionCookieValue
```



**Action reference**

**Parameters**

Name	Type	Required	Description
<a href="#">filter-definition</a>	Filter definition	N	A filter to reduce the volume of the response.
<a href="#">sort-definition</a>	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-bulk-objects>
    <row sco-id=integer type=allowedValue>
      <url>string</url>
      <name>string</name>
      <date-created>datetime</date-created>
      <date-end>datetime</date-end>
      <date-modified>datetime</date-modified>
      <description>datetime</description>
    </row>
    ...
  </report-bulk-objects>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-bulk-objects		Container	The entire list of SCOs on the server; or, if a filter is used, the entire list of SCOs that matches the filter.
row		Container	Details about one SCO.
	sco-id	Integer	The unique ID of the SCO.
	type	Allowed value	The type of SCO (see <a href="#">type</a> ).
url		String	The unique identifier of the training SCO, placed in the URL after the domain name.
name		String	The name assigned to the SCO.
date-created		Datetime	The date the SCO was created. For a meeting, the date and time the meeting starts.

**Action reference**

Element	Attribute	Type	Description
date-end		Datetime	If the SCO is a meeting or event, the date it ended.
date-modified		Datetime	The date the SCO was last updated.
description		String	The description of the SCO.

**Sample request**

```
http://example.com/api/xml?action=report-bulk-objects&filter-type=meeting
&filter-gt-date-created=2006-06-01
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-bulk-objects>
    <row sco-id="2006778715" type="meeting">
      <url>/online/</url>
      <name>Designing Online Courses</name>
      <date-created>2006-06-28T14:15:00.000-07:00</date-created>
      <date-end>2006-06-28T14:30:00.000-07:00</date-end>
      <date-modified>2006-07-13T14:57:54.150-07:00</date-modified>
    </row>
    ...
  </report-bulk-objects>
</results>
```

**See also**

[report-bulk-consolidated-transactions](#), [report-bulk-questions](#), [report-bulk-slide-views](#), [report-bulk-users](#)

## report-bulk-questions

**Availability**

Breeze 5

**Description**

Returns information about every quiz question in the account you are logged in to.

The response includes a combination of the quiz question, the answer, the ID of the user who answered, and the ID of the transaction.

This action returns all question-and-answer combinations in the account, unless you use a filter to limit the size of the response.

**Request URL**

```
http://server_name/api/xml
  ?action=report-bulk-questions
  &filter-definition=value
  &sort-definition=value
  &session=BreezeSessionCookieValue
```

**Action reference**

**Parameters**

Name	Type	Required	Description
<a href="#">filter-definition</a>	Filter definition	N	A filter to reduce the volume of the response.
<a href="#">sort-definition</a>	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the BREEZESESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-bulk-questions>
    <row transaction-id=integer score=integer principal-id=integer>
      <question>string</question>
      <response>string</response>
      <date-created>datetime</date-created>
    </row>
    ...
  </report-bulk-questions>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-bulk-questions		Container	The entire list of question-and-answer combinations that match the request.
row		Container	Details about one question-and-answer combination.
	transaction-id	Integer	The ID of the interaction between a user and a quiz.
	score	Integer	The score assigned to the question.
	principal-id	Integer	The ID of the user who answered or viewed the question.
question		String	The text of the question, which might be phrased as a statement.
response		String	The response the user chose or entered.
date-created		Datetime	The date and time the user answered the question.

**Sample request**

<https://example.com/api/xml?action=report-bulk-questions>

**Action reference****Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-bulk-questions>
    <row transaction-id="2006335803" score="10" principal-id="2006258745">
      <question>The capital of California is<1></question>
      <response>Sacramento</response>
      <date-created>2006-05-11T15:50:23.643-07:00</date-created>
    </row>
    <row transaction-id="2006335827" score="0" principal-id="2006258745">
      <question>The capital of California is<1></question>
      <response>san francisco</response>
      <date-created>2006-05-11T17:32:53.970-07:00</date-created>
    </row>
  </report-bulk-questions>
</results>
```

**See also**

[report-bulk-objects](#), [report-bulk-consolidated-transactions](#), [report-bulk-slide-views](#), [report-bulk-users](#)

**report-bulk-slide-views****Availability**

Breeze 5

**Description**

Returns information about each occasion on which a principal views a slide. The slide can be in any presentation in the account the current user belongs to.

Each slide view is a transaction. A *transaction* is an interaction between a user and any SCO on Adobe Connect. In this case, the transaction is between a user and a slide.

This action returns all occurrences of principals viewing slides in the account, unless you filter the response.

**Request URL**

```
http://server_name/api/xml
  ?action=report-bulk-slide-views
  &filter-definition=value
  &sort-definition=value
  &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
<a href="#">filter-definition</a>	Filter definition	N	A filter to reduce the volume of the response.
<a href="#">sort-definition</a>	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Action reference****Filters**

You can filter or sort the response on any element or attribute it contains.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-bulk-slide-views>
    <row transaction-id=integer principal-id=integer>
      <page>integer</page>
      <date-created>datetime</date-created>
    </row>
    ...
  </report-bulk-slide-views>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-bulk-slide-views		Container	The entire list of slide views that match the request.
row		Container	Details about one slide view.
	transaction-id	Integer	The ID of the interaction between the user and the slide.
	principal-id	Integer	The ID of the user who viewed the slide.
page		Integer	The page number of the slide in the presentation.
date-created		Datetime	The date and time the user viewed the slide.

**Sample request**

```
https://example.com/api/xml?action=report-bulk-slide-views
&filter-principal-id=123456
```

**Action reference****Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-bulk-slide-views>
    <row transaction-id="2006334916" principal-id="123456">
      <page>0</page>
      <date-created>2006-05-11T12:02:01.470-07:00</date-created>
    </row>
    <row transaction-id="2006334916" principal-id="123456">
      <page>0</page>
      <date-created>2006-05-11T12:02:01.487-07:00</date-created>
    </row>
    ...
  </report-bulk-slide-views>
</results>
```

**See also**

[report-bulk-objects](#), [report-bulk-questions](#), [report-bulk-consolidated-transactions](#), [report-bulk-users](#)

## report-bulk-users

**Availability**

Breeze 5

**Description**

Returns information about all users in an account. The difference between this call and `principal-list` is that `principal-list` returns both users and groups, while `report-bulk-users` returns only users.

The response from `report-bulk-users` can be quite large, especially if you use custom fields, so remember that you can filter and sort it. For example, the following call returns a list of all users who have the letters *Jo* in their name, in ascending order by name:

```
http://myserver.com/api/xml?action=report-bulk-users&sort-name=asc
&filter-like-name=Jo
```

If you pass `custom-fields=true`, by default `report-bulk-users` returns up to eight custom fields defined for users. If you have defined more than eight custom fields for users, `report-bulk-users` returns the first eight in the list in the Customize User Profile screen in Connect Central (at Administration > Users and Groups > Customize User Profile).

If you use Adobe Connect Server, you can set a value for `REPORT_MAX_CUSTOM_FIELDS` in the `custom.ini` file to have `report-bulk-users` return more than eight custom fields. You can use any value, but higher values risk a greater impact to database performance. You cannot change this setting on a Adobe Connect hosted account.

**Request URL**

```
http://server_name/api/xml
  ?action=report-bulk-users
  &custom-fields=boolean
  &filter-definition=value
  &sort-definition=value
  &session=BreezeSessionCookieValue
```

**Action reference**

**Parameters**

Name	Type	Required	Description
custom-fields	Boolean	N	Whether to return custom fields in the response. Returns up to eight custom fields. If true, the <code>manager</code> field is not returned in the response.
<a href="#">filter-definition</a>	Filter definition	N	A filter to reduce the volume of the response.
<a href="#">sort-definition</a>	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the <code>BREEZESSESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

You can use `filter-type` with `report-bulk-users` to filter the type of users returned (`user` or `guest`).

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-bulk-users>
    <row principal-id=integer type="string">
      <login>string</login>
      <name>string</name>
      <email>string</email>
      <manager>string</manager>
      .. any custom fields ..
    </row>
    ...
  </report-bulk-users>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	Top-level element for the response.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-bulk-users		Container	The entire list of users in the account.
row		Container	Details about one user in the account.
	principal-id	Integer	The ID of the user.
	type	String	The type of user, either <code>user</code> or <code>guest</code> .
login		String	The user's login ID, often an e-mail address.

**Action reference**

Element	Attribute	Type	Description
name		String	The full name of the user, concatenated from the user's first name and last name.
email		String	The user's e-mail address.
manager		String	The user's manager, also a registered user. Returned if a manager has been set for the user. Not returned if <code>custom-fields</code> is <code>true</code> in the request.

**Sample request**

```
https://example.com/api/xml?action=report-bulk-users&filter-like-name=john
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-bulk-users>
    <row principal-id="5417288" type="guest">
      <login>john@example.com</login>
      <name>John Owens</name>
      <email>john@example.com</email>
    </row>
    <row principal-id="5417255" type="user">
      <login>jsmith@example.com</login>
      <name>John Smith</name>
      <email>jsmith@example.com</email>
    </row>
    ...
  </report-bulk-users>
</results>
```

**See also**

[report-bulk-objects](#), [report-bulk-questions](#), [report-bulk-slide-views](#), [report-bulk-consolidated-transactions](#)

**report-course-status****Availability**

Breeze 4

**Description**

Returns summary information about a course, including the number of users who have passed, failed, and completed the course, as well as the current number of enrollees. The request requires the `sco-id` of a course.

Connect Central uses this call to display Course Status in the Summary report. This report is available at Training > Shared Training > *[course name]* > Reports > Summary.



**Action reference**

**Request URL**

```
http://server_name/api/xml
    ?action=report-course-status
    &sco-id=integer
    &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
sco-id	Integer	Y	The unique ID of the course for which you want summary information.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
  <results>
    <status code=allowedValue />
    <report-course-status total-course-completions=integer
      total-unique-course-completions=integer num-passed=integer
      num-failed=integer num-enrollees=integer />
    <date-last-taken>datetime</date-last-taken>
  </report-course-status>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-course-status		Container	Summary information about a course.
	total-course-completions	Integer	The total number of times users have completed the course, including passing scores, failing scores, and multiple attempts by the same user.
	total-unique-course-completions	Integer	The number of distinct users who have completed the course, including passing and failing scores but not multiple attempts by the same user.
	num-completed	Integer	The number of users who have completed the course, for courses that do not have a passing score.

**Action reference**

Element	Attribute	Type	Description
	num-passed	Integer	The number of users who have passed the course, for courses that have a passing score.
	num-failed	Integer	The number of users who have failed the course, for courses that have a passing score.
	num-enrollees	Integer	The number of users presently enrolled in the course.
date-last-taken		Datetime	The last time any user attempted the course but was not in server-side review mode.

**Sample request**

```
https://example.com/api/xml?action=report-course-status&sco-id=123456
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-course-status total-course-completions="1"
    total-unique-course-completions="1" num-completed="0"
    num-passed="1" num-failed="0" num-enrollees="4">
    <date-last-taken>2006-10-10T13:55:24.480-07:00</date-last-taken>
  </report-course-status>
</results>
```

**report-curriculum-taker****Availability**

Connect Enterprise 6

**Description**

Returns information about a user's progress in a curriculum.

The response includes a `row` element for each course in the curriculum, which has information such as access to the course, whether credit was granted, the user's score, the unique `url-path` to the course, and so on.

**Request URL**

```
http://server_name/api/xml
  ?action=report-curriculum-taker
  &user-id=integer
  &sco-id=integer
  &session=BreezeSessionCookieValue
```

**Action reference**

**Parameters**

Name	Type	Required	Description
user-id	Integer	Y	The ID of the user whose scores you want to check.
sco-id	Integer	Y	The unique ID of the curriculum for which you want a summary.
session	String	N	The value of the BREEZESESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-curriculum-taker>
    <sco transcript-id=integer path-type=allowedValue asset-id=integer
      sco-id=integer depth=integer folder-id=integer
      type=integer icon=allowedValue lang=allowedValue
      max-retries=integer source-sco-id=integer
      source-sco-type=allowedValue status=allowedValue score=integer
      certificate=integer max-score=integer attempts=integer>
    <access>allowedValue</access>
    <credit-granted>boolean</credit-granted>
    <name>string</name>
    <url-path>string</url-path>
    <date-modified>datetime</date-modified>
    <override>boolean</override>
  </sco>
</report-curriculum-taker>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-curriculum-taker		Container	Information about the user's performance in the entire curriculum.
sco		Container	Information about the user's work with one course or curriculum.
	transcript-id	Integer	The ID of the user's transcript for the course or curriculum.
	path-type	Allowed value	The learning path a user must take before attempting this course or curriculum (see <a href="#">path-type</a> for allowed values).

Action reference

Element	Attribute	Type	Description
	asset-id	Integer	The version of the course or curriculum the user attempted to complete. The <code>asset-id</code> is incremented each time the course or curriculum has new content uploaded.
	sco-id	Integer	The unique ID of the course or curriculum.
	depth	Integer	A course's level below the curriculum in the navigation hierarchy. For a curriculum, 0; for a course one level below the curriculum, 1.
	folder-id	Integer	The ID of the folder that contains the course or curriculum. For a course, the ID of a curriculum; for a curriculum, the ID of a user.
	type	Integer	The type of the course or curriculum (see <a href="#">type</a> for allowed values).
	icon	Allowed value	The type of icon that identifies the course or curriculum in Connect Central (see <a href="#">icon</a> for values).
	lang	Allowed value	The language associated with the course or curriculum (see <a href="#">lang</a> for values).
	max-retries	Integer	The maximum number of times a user can retake the course or curriculum. If a user can take the course 3 times, <code>max-retries</code> is 2.
	source-sco-id	Integer	The unique ID of the SCO used as a template for the course or curriculum.
	source-sco-type	Integer	The type of SCO used as a template for the course or curriculum (see <a href="#">type</a> for values).
	status	Allowed value	The status of the user's attempt to use the course or curriculum.  For courses, allowed values are <code>completed</code> , <code>incomplete</code> , <code>user-passed</code> , <code>user-failed</code> , and <code>not-attempted</code> .  For curriculums and folders, allowed values are <code>completed</code> and <code>incomplete</code> .
	score	Integer	The score the user earned on the course or curriculum.
	certificate	Integer	The ID of the user's certificate.
	max-score	Integer	The maximum score possible for the course or curriculum.
	attempts	Integer	The number of times the user has attempted the course or curriculum.
access		Allowed value	The level of access the user has to the course or curriculum (see <a href="#">access</a> for allowed values).
credit-granted		Boolean	A value indicating whether credit was granted for the course or curriculum.
name		String	The name of the learning object or curriculum.

**Action reference**

Element	Attribute	Type	Description
url-path		String	The part of the URL after the domain name that uniquely identifies the object on the server.
date-modified		Datetime	The date and time the SCO was last modified, in <a href="#">ISO 8601</a> format.
override		Boolean	A value indicating whether the transcript for the SCO has been adjusted.

**Sample request**

```
https://example.com/api/xml?action=report-curriculum-taker
&user-id=2006258748&sco-id=2006298444
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-curriculum-taker>
    <sco transcript-id="2006905613" path-type="prereq-none"
      asset-id="2006334911" sco-id="2006334909" depth="0"
      folder-id="2006258747" type="content" icon="producer"
      lang="en" max-retries="" source-sco-id="" source-sco-type=""
      status="user-failed" score="0" certificate="" max-score="0"
      attempts="5">
      <access>access-open</access>
      <credit-granted>>false</credit-granted>
      <name>Test Quiz</name>
      <url-path>/quiz</url-path>
      <date-created>2006-06-30T15:24:34.897-07:00</date-created>
      <date-modified>2006-05-16T15:22:25.703-07:00</date-modified>
      <date-taken>2006-06-30T15:24:34.897-07:00</date-taken>
      <override>>false</override>
    </sco>
  </report-curriculum-taker>
</results>
```

**report-meeting-attendance**

**Availability**

Breeze 4

**Description**

Returns a list of users who attended a Adobe Connect meeting. The data is returned in `row` elements, one for each person who attended. If the meeting hasn't started or had no attendees, the response contains no rows. The response does not include meeting hosts or users who were invited but did not attend.

To call `report-meeting-attendance`, you must have `publish`, `mini-host`, or `host` permission on the meeting (see [permission-id](#) for details).

**Action reference**

**Request URL**

```
http://server_name/api/xml
    ?action=report-meeting-attendance
    &sco-id=integer
    &filter-definition=value
    &sort-definition=value
    &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
sco-id	Integer	Y	The unique ID of a meeting.
<a href="#">filter-definition</a>	Filter definition	N	A filter to reduce the volume of the response.
<a href="#">sort-definition</a>	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-meeting-attendance>
    <row transcript-id=integer sco-id=integer principal-id=integer
      answered-survey=boolean>
      <login>string</login>
      <session-name>string</session-name>
      <sco-name>string</sco-name>
      <date-created>datetime</date-created>
      <date-end>datetime</date-end>
      <participant-name>string</participant-name>
    </row>
    ...
  </report-meeting-attendance>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-meeting-attendance		Container	The entire list of attendees for the meeting.
row		Container	Data about one meeting attendee.
	transcript-id	Integer	The ID of the meeting transcript.

**Action reference**

Element	Attribute	Type	Description
	sco-id	Integer	The unique ID of the meeting.
	principal-id	Integer	The ID of the principal who attended the meeting.
	answered-survey	Boolean	Whether the meeting participant responded to a meeting poll. If 0 or false, the meeting did not have a poll or the participant did not respond (if 1 or true, the opposite). This value is updated when the poll is closed.
login		String	The meeting attendee's login name.
session-name		String	The name of the user who entered the meeting room, creating a session.
sco-name		String	The name of the meeting.
date-created		Datetime	The date the meeting was created.
date-end		Datetime	The date the meeting ended.
participant-name		String	The name of the meeting attendee as registered with the server.

**Sample request**

```
https://example.com/api/xml?action=report-meeting-attendance
&sco-id=2006778715
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-meeting-attendance>
    <row transcript-id="2006778723" sco-id="2006778715"
      principal-id="2006258745" answered-survey="0">
      <login>joy@acme.com</login>
      <session-name>Joy Smith</session-name>
      <sco-name>Designing Online Courses</sco-name>
      <date-created>2006-06-28T14:35:21.307-07:00</date-created>
      <date-end>2006-06-28T15:09:05.447-07:00</date-end>
      <participant-name>Joy Smith</participant-name>
    </row>
  </report-meeting-attendance>
</results>
```

**report-meeting-concurrent-users**

**Availability**

Breeze 4

**Description**

Returns the maximum number of users in Adobe Connect meetings concurrently in the last 30 days, and the number of times the maximum has been reached. The maximum is the peak number of users in any meetings at a single moment, whether one meeting, multiple concurrent meetings, or multiple overlapping meetings.

**Action reference**

You can change the time period to a period greater than 30 days by adding a `length` parameter, for example, `length=120`.

The maximum number of users (`max-users`) is determined by the account license and applies to the server overall, not to a specific meeting. This action also returns the number of times in the current month the maximum has been reached (`max-participants-freq`).

**Request URL**

```
http://server_name/api/xml
    ?action=report-meeting-concurrent-users
    &length=integer
    &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
length	Integer	N	The number of days in the time period to check for concurrent meeting usage. Use a value greater than 30. The default value is 30.
session	String	N	The value of the BREEZESESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<results>
  <status code=allowedValue />
  <report-meeting-concurrent-users max-users=integer
    max-participants-freq=integer />
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-meeting-concurrent-users		Empty, with attributes	Information about the peak number of users in meetings at the same moment.
	max-users	Integer	The peak number of users in meetings at the same moment (either a single meeting or concurrent meetings) during the time period.
	max-participants-freq	Integer	The number of times the maximum has been reached in the time period.



**Action reference****Sample request**

```
https://example.com/api/xml?action=report-meeting-concurrent-users
```

**Sample response**

```
<results>
  <status code="ok" />
  <report-meeting-concurrent-users max-users="400"
    max-participants-freq="1" />
</results>
```

**report-meeting-session-users****Availability**

Adobe Connect 8.1

**Description**

Provides information about all the user sessions for an Adobe Connect meeting session. A user session is created when a participant enters a meeting session. As more participants join the meeting, they join the meeting session. The user session ends when the user leaves the meeting session. When a new participant enters an empty meeting, a new meeting session and new user session is started.

**Request URL**

```
http://server_name/api/xml
  ?action=report-meeting-session-users
  &sco-id=integer
  &asset-id=integer
  &filter-definition=value
  &sort-definition=value
  &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
sco-id	Integer	Y	The ID of a meeting for which you want user session information.
asset-id	Integer	Y	The ID of the meeting session. As returned by report-meetings-sessions.
filter-definition	Filter definition	N	A filter to reduce the volume of the response.
sort-definition	Filter definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the BREEZESESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

**Action reference**

**Response structure**

```
<?xml version="1.0" encoding="utf-8"?>
<results>
  <status code="ok"/>
  <report-meeting-session-users>
    <row principal-id=integer>
      <principal-name>string</principal-name>
      <date-created>datetime</date-created>
      <date-end>datetime</date-end>
    </row>
    ...
  </report-meeting-session-users>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results that the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see status).
report-meeting-session-users		Container	The list of user-sessions for the meeting session.
row		Container	The information for one user-session.
	principal-id		Unique ID of a user/participant.
principal-name		String	Name of a user/participant.
date-created		Datetime	The date and time the user-session was created when the participant entered the meeting session.
date-end		Datetime	The date and time the user-session ended when the participant exited the meeting session.

**Sample request**

```
https://example.com/api/xml?action=report-meeting-session-users&sco-id=2006811328&asset-id=446653455
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8"?>
<results>
  <status code="ok"/>
  <report-meeting-session-users>
    <row principal-id="52904333">
      <principal-name>John Doe</principal-name>
      <date-created>2011-08-30T15:46:46.190-04:00</date-created>
      <date-end>2011-08-30T16:51:30.950-04:00</date-end>
    </row>
  </report-meeting-session-users>
</results>
```

## report-meeting-sessions

### Availability

Breeze 4

### Description

Provides information about all the sessions of a Adobe Connect meeting. A *session* is created when a participant enters an empty meeting. As more participants join the meeting, they join the session. The session ends when all attendees leave the meeting. When a new participant enters the now-empty meeting, a new session starts. For example, a recurring weekly meeting has a session each week when the meeting is held.

You can call `report-meeting-sessions` on past meetings, active meetings, or future meetings, but future meetings are not likely to have sessions.

### Request URL

```
http://server_name/api/xml
?action=report-meeting-sessions
&sco-id=integer
&filter-definition=value
&sort-definition=value
&session=BreezeSessionCookieValue
```

### Parameters

Name	Type	Required	Description
sco-id	Integer	Y	The ID of a meeting for which you want session information.
<a href="#">filter-definition</a>	Filter definition	N	A filter to reduce the volume of the response.
<a href="#">sort-definition</a>	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

### Filters

You can filter or sort the response on any element or attribute it contains.

### Response structure

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-meeting-sessions>
    <row sco-id=integer asset-id=integer version=integer
      num-participants=integer>
      <date-created>datetime</date-created>
      <date-end>datetime</date-end>
    </row>
    ...
  </report-meeting-sessions>
</results>
```

**Action reference****Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-meeting-sessions		Container	The entire list of sessions for the meeting.
row		Container	Information about one session.
	sco-id	Integer	The unique ID of the meeting.
	asset-id	Integer	The unique ID of the session.
	version	Integer	A sequential ID for the session, starting at 1.
	num-participants	Integer	The number of participants in the meeting, other than the host.
date-created		Datetime	The date and time the session was created, when the participant entered the meeting room.
date-end		Datetime	The date and time the session ended, when the participant left the meeting room.

**Sample request**

```
https://example.com/api/xml?action=report-meeting-sessions
&sco-id=2006811328
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-meeting-sessions>
    <row sco-id="2006811328" asset-id="2006811333" version="1"
      num-participants="1">
      <date-created>2006-06-29T11:46:52.210-07:00</date-created>
      <date-end>2006-06-29T13:34:43.410-07:00</date-end>
    </row>
  </report-meeting-sessions>
</results>
```

**report-meeting-summary****Availability**

Breeze 4

**Description**

Returns summary information about a specific Adobe Connect meeting. The results indicate how many users were invited, how many invited participants and guests attended, and other information about the meeting.

To use `report-meeting-summary`, you need `publish`, `host`, or `mini-host` permission on the meeting. With one of these permissions, you can run `report-meeting-summary` on a current, completed, or future meeting. The results are most useful for a completed meeting.

**Action reference**

A meeting might be recurring (for example, a weekly staff meeting) and have an occurrence each time the meeting is held. If the meeting is recurring, the statistics returned by `report-meeting-summary` are cumulative, applying to all occurrences of the meeting, not just the latest one.

**Request URL**

```
http://server_name/api/xml
    ?action=report-meeting-summary
    &sco-id=integer
    &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
sco-id	Integer	Y	The unique ID of a meeting for which you have <code>publish</code> or <code>host</code> permission.
session	String	N	The value of the <code>BREEZESSESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-meeting-summary num-unique-meetings=integer peak-users=integer
    num-invitees=integer num-invitees-attended=integer ispublic=boolean
    num-guests-attended=integer />
</results>
```

**Returned XML elements**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<code>status</code>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <code>status</code> ).
report-meeting-summary		Empty, with attributes	Details about the meeting or meeting series.
	num-unique-meetings	Integer	The number of occurrences of a recurring meeting.
	peak-users	Integer	The highest number of participants in the meeting room at one time, during any one meeting occurrence.
	num-invitees	Integer	The number of users who were invited.

**Action reference**

Element	Attribute	Type	Description
	num-invitees-attended	Integer	The number of invited users who attended.
	ispublic	Boolean	Whether the meeting is public and guests can enter automatically (if 1 or true), or private and must wait for permission (if 0 or false).
	num-guests-attended	Integer	The number of participants who entered the meeting room as guests rather than as registered attendees.

**Sample request**

```
https://example.com/api/xml?action=report-meeting-summary&sco-id=2006334033
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-meeting-summary num-unique-meetings="1" peak-users="1"
    num-invitees="1" num-invitees-attended="1" ispublic="1"
    num-guests-attended="0">
    <most-recent-session>
      2006-06-28T15:11:15.133-07:00
    </most-recent-session>
  </report-meeting-summary>
</results>
```

**report-my-courses****Availability**

Breeze 4

**Description**

Provides information about each course the current user is or was enrolled in.

The returned courses include future courses, past courses, and courses the user is presently taking. The list of courses can be quite large, so remember to use a filter to reduce the response.

Each course has a `permission-id` that shows the level of access the user has to the course. For example, the access might be `view`, `publish`, or `manage`.

**Request URL**

```
http://server_name/api/xml
  ?action=report-my-courses
  &filter-definition=value
  &sort-definition=value
  &session=BreezeSessionCookieValue
```

**Action reference**

**Parameters**

Name	Type	Required	Description
<a href="#">filter-definition</a>	Filter definition	N	A filter to reduce the volume of the response.
<a href="#">sort-definition</a>	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the BREEZESESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <my-courses>
    <course sco-id=integer type="content" icon="course"
      permission-id=allowedValue>
      <name>string</name>
      <description>string</description>
      <url>string</url>
      <date-created>datetime</date-created>
      <date-modified>datetime</date-modified>
      <date-begin>datetime</date-begin>
      <url-path>string</url-path>
      <expired>boolean</expired>
      <completed>boolean</completed>
    </course>
    ...
  </my-courses>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
my-courses		Container	Information about all courses the user is enrolled in.
course		Container	Information about one course the user is enrolled in.
	sco-id	Integer	The unique ID of the course.
	type	Allowed value	The type of the course (for allowed values, see <a href="#">type</a> ).
	icon	Allowed value	The type of icon that identifies the course in the user interface. For a course, always <code>course</code> .
	permission-id	Allowed value	The level of permission the user has on the course (see <a href="#">permission-id</a> for values).
name		String	The name of the course.

**Action reference**

Element	Attribute	Type	Description
url		String	The URL at which a user can reach the course on the server. Includes the domain name and the course identifier.
date-created		Datetime	The date and time the course was created.
date-modified		Datetime	The date and time the course was last modified.
date-begin		Datetime	The date and time the course is available for users to start.
date-end		Datetime	The date and time the course closes.
url-path		String	The part of the course URL that is the course identifier, after the domain name.
expired		Boolean	Whether the course has expired ( <code>true</code> if it has, <code>false</code> if it has not).
completed		Boolean	Whether the user has completed the course ( <code>true</code> if yes, <code>false</code> if no).

**Sample request**

```
https://example.com/api/xml?action=report-my-courses
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <my-courses>
    <course sco-id="2006298431" type="content" icon="course"
      permission-id="view">
      <name>Test Course</name>
      <url>example.com/test</url>
      <date-created>2006-05-03T10:21:46.810-07:00</date-created>
      <date-modified>2006-05-03T10:22:30.803-07:00</date-modified>
      <date-begin>2006-05-03T10:15:00.000-07:00</date-begin>
      <url-path>/test/</url-path>
      <expired>false</expired>
      <completed>false</completed>
    </course>
  </my-courses>
</results>
```

**report-my-events****Availability**

Breeze 5

**Description**

Provides information about each event the current user has attended or is scheduled to attend. The user can be either a host or a participant in the event. The events returned are those in the user's `my-events` folder.

To obtain information about all events on your Adobe Connect Server or in your Adobe Connect hosted account, call `sco-shortcuts` to get the `sco-id` of the `events` folder. Then, call `sco-contents` with the `sco-id` to list all events.



**Action reference**

**Request URL**

```
http://server_name/api/xml
    ?action=report-my-events
    &filter-definition=value
    &sort-definition=value
    &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
<a href="#">filter-definition</a>	Filter definition	N	A filter to reduce the volume of the response.
<a href="#">sort-definition</a>	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <my-events>
    <event sco-id=integer type="event" icon="event"
      permission-id=allowedValue>
      <name>string</name>
      <domain-name>string</domain-name>
      <url-path>string</url-path>
      <date-begin>datetime</date-begin>
      <date-end>datetime</date-end>
      <expired>boolean</expired>
      <duration>datetime/duration>
    </event>
    ...
  </my-events>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
my-events		Container	The entire list of events the user is or has been registered for.
event		Container	Information about one event.
	sco-id	Integer	The unique ID of the event.
	type	Allowed value	The type of the object. For an event, always event.
	icon	Allowed value	An icon identifying the object. For an event, always event.

**Action reference**

Element	Attribute	Type	Description
	permission-id	Allowed value	The permission the user has for the event (see <a href="#">permission-id</a> for values).
name		String	The name of the event.
domain-name		String	The domain name of the Adobe Connect server, which comes after <a href="#">http://</a> (or <a href="#">https://</a> ) and before the unique event name in the event URL.
url-path		String	The unique event name, which comes after the domain name in the event URL.
date-begin		Date	The date the event begins, in <a href="#">ISO 8601</a> format.
date-end		Date	The date the event ends, in <a href="#">ISO 8601</a> format.
expired		Boolean	A value indicating whether the event has ended. If the event is currently underway, the value is <code>false</code> .
duration		Time	The amount of time the event is scheduled to last. Uses the time portion of an <a href="#">ISO 8601</a> date format.

**Sample request**

`https://example.com/api/xml?action=report-my-events`

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <my-events>
    <event sco-id="2006334107" type="event" icon="event"
      permission-id="host">
      <name>Meet the Famous Author</name>
      <domain-name>example.com</domain-name>
      <url-path>/author/</url-path>
      <date-begin>2006-05-12T18:00:00.000-07:00</date-begin>
      <date-end>2006-05-12T20:00:00.000-07:00</date-end>
      <expired>true</expired>
      <duration>02:00:00.000</duration>
    </event>
  </my-events>
</results>
```

**report-my-meetings**

**Availability**

Breeze 4

**Description**

Provides information about all Adobe Connect meetings for which the user is a host, invited participant, or registered guest. The meeting can be scheduled in the past, present, or future.

**Action reference**

**Request URL**

```
http://server_name/api/xml
    ?action=report-my-meetings
    &filter-definition=value
    &sort-definition=value
    &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
<a href="#">filter-definition</a>	Filter definition	N	A filter to reduce the volume of the response.
<a href="#">sort-definition</a>	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <my-meetings>
    <meeting sco-id=integer type="meeting" icon="meeting"
      permission-id=allowedValue active-participants=integer>
      <name>string</name>
      <description>string</description>
      <domain-name>domain</domain-name>
      <url-path>url</url-path>
      <date-begin>date</date-begin>
      <date-end>date</date-end>
      <expired>boolean</expired>
      <duration>time</duration>
    </meeting>
    ...
  </my-meetings>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
my-meetings		Container	Information about all meetings the user is, or has been, invited to.
meeting		Container	Details about one of the user's meetings.
	sco-id	Integer	The unique ID of the meeting.

**Action reference**

Element	Attribute	Type	Description
	type	Allowed value	The type of the object returned (for this call, always <code>meeting</code> ).
	icon	Allowed value	The icon that visually identifies the meeting in Connect Central (for this call, always <code>meeting</code> ).
	permission-id	Allowed value	The level of permission the user has to the meeting (see <a href="#">permission-id</a> for values).
	active-participants	Integer	The number of participants the meeting currently has, including hosts and presenters.
name		String	The name of the meeting.
domain-name		String	The domain name portion of the URL to the meeting room.
url-path		String	The part of the meeting room URL that identifies the meeting and comes after the domain name.
date-begin		Datetime	The date and time the meeting begins (or has begun).
date-end		Datetime	The date and time the meeting ends (or has ended).
expired		Boolean	Whether the meeting has ended ( <code>true</code> if it has, <code>false</code> if it has not).
duration		Time	The actual length of time of the meeting. This may be longer or shorter than the time the meeting was scheduled for.

**Sample request**

```
https://example.com/api/xml?action=report-my-meetings
```

**Action reference****Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <my-meetings>
    <meeting sco-id="2006334033" type="meeting" icon="meeting"
      permission-id="host" active-participants="0">
      <name>How to Write a Novel</name>
      <domain-name>example.com</domain-name>
      <url-path>/novel/</url-path>
      <date-begin>2006-05-11T11:30:00.000-07:00</date-begin>
      <date-end>2006-05-11T12:30:00.000-07:00</date-end>
      <expired>true</expired>
      <duration>01:00:00.000</duration>
    </meeting>
    <meeting sco-id="2006743452" type="meeting" icon="meeting"
      permission-id="host" active-participants="0">
      <name>Intro to Film</name>
      <domain-name>example.com</domain-name>
      <url-path>/film/</url-path>
      <date-begin>2006-06-09T14:00:00.000-07:00</date-begin>
      <date-end>2006-06-09T20:00:00.000-07:00</date-end>
      <expired>true</expired>
      <duration>06:00:00.000</duration>
    </meeting>
  </my-meetings>
</results>
```

**report-my-training****Availability**

Connect Enterprise 6

**Description**

Returns a list of all courses and curriculums a user or group is enrolled in. If you do not use a `principal-id`, the list is for the current user. If you add a `principal-id`, the list is for the principal you specify.

The response contains a list of row elements. In the list, courses have the attributes `type=content` and `icon=course`, while curriculums have `type=curriculum` and `icon=curriculum`.

**Request URL**

```
http://server_name/api/xml
?action=report-my-training
&principal-id=integer
&filter-definition=value
&sort-definition=value
&session=BreezeSessionCookieValue
```

**Action reference**

**Parameters**

Name	Type	Required	Description
principal-id	Integer	N	The unique ID of a user or group whose courses and curriculums you want to list. If you do not specify a value, the response is for the current user.
<a href="#">filter-definition</a>	Filter definition	N	A filter to reduce the volume of the response.
<a href="#">sort-definition</a>	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-my-training>
    <row sco-id=integer type=allowedValue icon=allowedValue max-retries=integer
      permission-id=allowedValue transcript-id=integer attempts=integer>
      <name>string</name>
      <url>string</url>
      <date-created>datetime</date-created>
      <date-modified>datetime</date-modified>
      <date-begin>datetime</date-begin>
      <url-path>string</url-path>
      <expired>boolean</expired>
      <completed>boolean</completed>
    </row>
    ...
  </report-my-training>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-my-training		Container	The entire list of courses and curriculums the user is enrolled in.
row		Container	Information about one course or curriculum the user is enrolled in.
	sco-id	Integer	The unique ID of the course or curriculum.
	type	Allowed value	The type of the object (see <a href="#">type</a> for allowed values).
	icon	Allowed value	The icon that identifies the object in Connect Central (see <a href="#">icon</a> for allowed values). If <a href="#">type</a> is content, the <a href="#">icon</a> value describes the content.

**Action reference**

Element	Attribute	Type	Description
	max-retries	Integer	The allowed number of attempts that the course can be retaken.
	permission-id	Allowed value	The permission the principal has on the object (see <a href="#">permission-id</a> for allowed values).
	transcript-id	Integer	The ID of the course transcript.
	attempts	Integer	The number of times the user has tried to complete the course.
name		String	The name of the course or curriculum.
description		String	The course or curriculum description.
url		String	The part of the URL to the course or curriculum that includes the domain name and unique name, without <code>http://</code> or <code>https://</code> .
date-created		Datetime	The date and time the course or curriculum was created.
date-modified		Datetime	The date and time the course or curriculum was last modified.
date-begin		Datetime	The start date and time of the course or curriculum, either past or future.
date-end		Datetime	The end date or time of the course or curriculum, either past or future.
sco-tag		String	A non-unique identifier for the course or curriculum as shown in the user interface, for example, ECON101.
url-path		String	The unique name of the course or curriculum in its URL.
expired		Boolean	A value indicating whether the end date of the course or curriculum has passed ( <code>true</code> if it has, <code>false</code> if not).
completed		Boolean	A value indicating whether the user or group has completed the course.
tr-status		String	Whether the user has attempted to take the course ( <code>attempted</code> ) or not ( <code>not-attempted</code> ).

**Sample request**

```
https://example.com/api/xml?action=report-my-training
&principal-id=2006258745
```

**Action reference****Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-my-training>
    <row sco-id="2006298431" type="content" icon="course" max-retries=""
      permission-id="view" transcript-id="" attempts="0">
      <name>Intro to Psychology</name>
      <url>example.com/psychology/</url>
      <date-created>2006-05-03T10:21:46.810-07:00</date-created>
      <date-modified>2006-05-03T10:22:30.803-07:00</date-modified>
      <date-begin>2006-05-03T10:15:00.000-07:00</date-begin>
      <url-path>/psychology/</url-path>
      <expired>>false</expired>
      <completed>>true</completed>
    </row>
    <row sco-id="2006745669" type="curriculum" icon="curriculum"
      permission-id="view">
      <name>A Day in the Life</name>
      <url>example.com/day/</url>
      <date-created>2006-06-12T14:47:59.903-07:00</date-created>
      <date-modified>2006-06-12T14:47:59.903-07:00</date-modified>
      <date-begin>2006-06-12T14:45:00.000-07:00</date-begin>
      <url-path>/day/</url-path>
      <expired>>false</expired>
      <completed>>false</completed>
      <tr-status>not-attempted</tr-status>
    </row>
  </report-my-training>
</results>
```

**report-quiz-interactions****Availability**

Breeze 4

**Description**

Provides information about all the interactions users have had with a certain quiz. An *interaction* identifies all answers one user makes to one quiz question. If a user answers the same question more than once, all answers are part of the same interaction and have the same *interaction-id*.

This report provides information about every answer that any user has ever given to questions on a quiz. You can filter the response to make it more meaningful, using any allowed filters. For example, you can request all answers a certain user has given:

```
https://example.com/api/xml?action=report-quiz-interactions
&sco-id=2006334909&filter-like-name=Joy%20Smith
```

Or, you can request only a certain user's answers to a specific question:

```
https://example.com/api/xml?action=report-quiz-interactions
&sco-id=2006334909&filter-name=Joy%20Smith
&filter-like-description=What is the capital of California
```



**Action reference**

**Request URL**

```
http://server_name/api/xml
    ?action=report-quiz-interactions
    &sco-id=integer
    &filter-definition=value
    &sort-definition=value
    &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
sco-id	Integer	Y	The unique ID of a presentation or course that contains a quiz.
filter-definition	Filter definition	N	A filter to reduce the volume of the response.
sort-definition	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

**Response structure**

```
<results>
  <status code=allowedValue />
  <report-quiz-interactions>
    <row display-seq=integer transcript-id=integer interaction-id=integer
      sco-id=integer score=integer>
      <name>string</name>
      <sco-name>string</sco-name>
      <date-created>datetime</date-created>
      <description>string</description>
      <response>integer</response>
    </row>
    ...
  </report-quiz-interactions>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-quiz-interactions		Container	Information about all interactions all users have had with the quiz.
row		Container	Information about one user, one quiz question, and one answer. Multiple row elements can be part of the same interaction.
	display-seq	Integer	The sequence number of this question in the quiz.

**Action reference**

Element	Attribute	Type	Description
	transcript-id	Integer	The ID of one user's attempt to take a quiz, with one user, one attempt at a quiz, and multiple questions and answers. Each time the user takes the quiz, the transcript-id changes.
	interaction-id	Integer	The ID of all answers one user makes to one quiz question.
	sco-id	Integer	The unique ID of a presentation or course that contains the quiz.
	score	Integer	The user's score for this question.
name		String	The name of the user.
sco-name		Integer	The name of the presentation or course that contains the quiz.
date-created		Datetime	The date the presentation or course was created.
description		String	The quiz question the user answered.
response		String	The response the user gave.

**Sample request**

```
https://example.com/api/xml?action=report-quiz-interactions
&sco-id=2006334909&filter-name=Joy Smith
&filter-like-description=governor
```

**Action reference****Sample request**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
<status code="ok" />
  <report-quiz-interactions>
    <row display-seq="2" transcript-id="2006335803"
      interaction-id="2006334914" sco-id="2006334909" score="10">
      <name>Joy Smith</name>
      <sco-name>California Quiz</sco-name>
      <date-created>2006-05-11T15:50:23.643-07:00</date-created>
      <description>
        The governor of California is a former actor.
      </description>
      <response>true</response>
    </row>
    <row display-seq="2" transcript-id="2006335827"
      interaction-id="2006334914" sco-id="2006334909" score="0">
      <name>Joy Smith</name>
      <sco-name>California Quiz</sco-name>
      <date-created>2006-05-11T17:32:53.970-07:00</date-created>
      <description>
        The governor of California is a former actor.
      </description>
      <response>>false</response>
    </row>
    <row display-seq="2" transcript-id="2006335954"
      interaction-id="2006334914" sco-id="2006334909" score="10">
      <name>Joy Smith</name>
      <sco-name>California Quiz</sco-name>
      <date-created>2006-05-12T11:55:24.940-07:00</date-created>
      <description>
        The governor of California is a former actor.
      </description>
      <response>true</response>
    </row>
  </report-quiz-interactions>
</results>
```

**report-quiz-question-answer-distribution****Availability**

Breeze 4

**Description**

Returns information about the number of users who chose a specific answer to a quiz question. The combination of one quiz question and all of one user's answers to it is called an *interaction*. If the user answers the question more than once, all answers are part of the same interaction and have the same *interaction-id*.

Call [report-quiz-interactions](#) to determine an *interaction-id* to specify in the request. The *interaction-id* does not correspond to the question number in the quiz (for example, question 1, question 2, and so on).

**Action reference**

**Request URL**

```
http://server_name/api/xml
    ?action=report-quiz-question-answer-distribution
    &interaction-id=integer
    &sco-id=integer
    &filter-definition=value
    &sort-definition=value
    &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
interaction-id	Integer	N	The ID that describes all of one user's responses to one quiz question.
sco-id	Integer	Y	The unique ID of a presentation or course that contains a quiz.
filter-definition	Filter definition	N	A filter to reduce the volume of the response.
sort-definition	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-quiz-question-answer-distribution>
    <row display-seq=integer interaction-id=integer score=integer
      asset-id=integer num-selected=integer>
      <response>string</response>
    </row>
    ...
  </report-quiz-question-answer-distribution>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-quiz-question-answer-distribution		Container	The list of questions and answers.
row		Container	Information about one user, one question, and one answer.
	display-seq	Integer	The sequence number of the question in the quiz.

**Action reference**

Element	Attribute	Type	Description
	interaction-id	Integer	The ID of all of one user's responses to one quiz question. If the user answers the question multiple times, all answers have the same interaction-id.
	score	Integer	The score the user earned on the question.
	asset-id	Integer	The ID of the version of the quiz in which the user answered the question. The asset-id changes when you upload a new content version.
	num-selected	Integer	In multiple choice or true/false quiz questions, the sequence number of the answer selected.
response		String	The response the user gave to the question.

**Sample request**

```
https://example.com/api/xml
?action=report-quiz-question-answer-distribution&sco-id=2006334909
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-quiz-question-answer-distribution>
    <row display-seq="1" interaction-id="2006334913" score="0"
      asset-id="2006334911" num-selected="1">
      <response>san francisco</response>
    </row>
    <row display-seq="1" interaction-id="2006334913" score="10"
      asset-id="2006334911" num-selected="2">
      <response>Sacramento</response>
    </row>
    <row display-seq="2" interaction-id="2006334914" score="0"
      asset-id="2006334911" num-selected="1">
      <response>>false</response>
    </row>
    ...
  </report-quiz-question-answer-distribution>
</results>
```

**report-quiz-question-distribution****Availability**

Breeze 4

**Action reference**

**Description**

Returns information about the number of correct and incorrect answers to the questions on a quiz. This call can help you determine how a group responded to a quiz question overall.

Because this call returns information about all the questions on a quiz, you may want to filter the results for a specific question or group of questions.

**Request URL**

```
http://server_name/api/xml
    ?action=report-quiz-question-distribution
    &sco-id=integer
    &filter-definition=value
    &sort-definition=value
    &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
sco-id	Integer	Y	The unique ID of a presentation that contains a quiz.
<a href="#">filter-definition</a>	Filter definition	N	A filter to reduce the volume of the response.
<a href="#">sort-definition</a>	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the BREEZESESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-quiz-question-distribution>
    <row display-seq=integer interaction-id=integer num-correct=integer
      num-incorrect=integer total-responses=integer
      percentage-correct=integer score=integer>
      <name>string</name>
      <description>string</description>
    </row>
    ...
  </report-quiz-question-distribution>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Action reference**

Element	Attribute	Type	Description
report-quiz-question-distribution		Container	Information about all the questions on a quiz.
row		Container	Information about one question on the quiz.
	display-seq	Integer	The sequence in the quiz in which this question falls.
	interaction-id	Integer	The ID of the quiz question.
	num-correct	Integer	The number of correct answers to this question.
	num-incorrect	Integer	The number of incorrect answers to this question.
	total-responses	Integer	The total number of responses to this question.
	percentage-correct	Integer	The percentage of the total responses that were correct.
	score	Integer	The score assigned to the quiz question.
name		String	The name of the quiz question, defined when the question was created in Quiz Manager.
description		String	The definition of the quiz question, defined when the question was created in Quiz Manager.

**Sample request**

```
https://example.com/api/xml?action=report-quiz-question-distribution
&sco-id=2006334909&filter-like-description=The capital of California
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-quiz-question-distribution>
    <row display-seq="1" interaction-id="2006334913" num-correct="2"
      num-incorrect="1" total-responses="3" percentage-correct="66"
      score="10">
      <name>The capital of California is<1></name>
      <description>The capital of California is<1></description>
    </row>
  </report-quiz-question-distribution>
</results>
```

**report-quiz-question-response**

**Availability**

Breeze 4

**Action reference**

**Description**

Provides a list of answers that users have given to questions on a quiz.

Without filtering, this action returns all answers from any user to any question on the quiz. However, you can filter the response for a specific user, interaction, or answer (see the filter syntax at [filter-definition](#)).

An *interaction* is a combination of one user and one question. If the user answers the same question more than once, all answers are part of the same interaction and have the same `interaction-id`.

**Request URL**

```
http://server_name/api/xml
    ?action=report-quiz-question-response
    &sco-id=integer
    &filter-definition=value
    &sort-definition=value
    &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
sco-id	Integer	Y	The unique ID of a presentation that contains a quiz.
<a href="#">filter-definition</a>	Filter definition	N	A filter to reduce the volume of the response.
<a href="#">sort-definition</a>	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-quiz-question-response>
    <row principal-id=integer interaction-id=string>
      <user-name>string</user-name>
      <response>string</response>
      <date-created>datetime</date-created>
    </row>
  </report-quiz-question-response>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).



**Action reference**

Element	Attribute	Type	Description
report-quiz-question-response		Container	Information about all responses to all questions on the quiz.
row		Container	Information about one response.
	principal-id	Integer	The ID of the user who answered the quiz question.
	interaction-id	Integer	The ID of one response to one question.
user-name		String	The name of the user as registered on the server.
response		String	The user's response to the question, including a word or phrase, true, false, or a letter choice.
date-created		Datetime	The date and time the user responded.

**Sample request**

```
https://example.com/api/xml?action=report-quiz-question-response
&sco-id=2006334909&filter-interaction-id=2006334913
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-quiz-question-response>
    <row principal-id="2006258745" interaction-id="2006334913">
      <user-name>Joy Smith</user-name>
      <response>Sacramento</response>
      <date-created>2006-05-11T15:50:23.643-07:00</date-created>
    </row>
    <row principal-id="2006258745" interaction-id="2006334913">
      <user-name>Joy Smith</user-name>
      <response>san francisco</response>
      <date-created>2006-05-11T17:32:53.970-07:00</date-created>
    </row>
    <row principal-id="2006258745" interaction-id="2006334913">
      <response>Sacramento</response>
      <date-created>2006-05-12T11:55:24.940-07:00</date-created>
    </row>
  </report-quiz-question-response>
</results>
```

**report-quiz-summary****Availability**

Breeze 4

**Description**

Provides a summary of data about a quiz, including the number of times the quiz has been taken; average, high, and low scores; and other information.

**Action reference**

**Request URL**

```
http://server_name/api/xml
    ?action=report-quiz-summary
    &sco-id=integer
    &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
sco-id	Integer	Y	The unique ID of a presentation that contains a quiz.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-quiz-summary>
    <row num-questions=integer average-score=integer low-score=integer
      high-score=integer numtaken=integer numdistincttaken=integer
      maxpossiblescore=integer asset-id=integer />
  </report-quiz-summary>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-quiz-summary		Container	Contains information about the quiz.
row		Empty, with attributes	Summary information about the quiz. Can return more than one <code>row</code> element if the <code>maxpossiblescore</code> is different for different transcripts.
	num-questions	Integer	The number of questions on the quiz.
	average-score	Integer	The average score, across all users who have taken the quiz.
	low-score	Integer	The lowest score a user has received on the quiz.
	high-score	Integer	The highest score a user has received on the quiz.
	numtaken	Integer	The total number of times the quiz has been taken.

**Action reference**

Element	Attribute	Type	Description
	numdistincttaken	Integer	The number of times the quiz has been taken by distinct principals. If a principal takes the quiz more than once, only one time is counted.
	maxpossiblescore	Integer	The highest possible score on the quiz.
	asset-id	Integer	The ID of the latest version of the SCO uploaded to the server.

**Sample request**

```
https://server.com/api/xml?action=report-quiz-summary&sco-id=2006123456
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
<status code="ok" />
  <report-quiz-summary>
    <row num-questions="2" average-score="0" low-score="0" high-score="0"
      numtaken="1" numdistincttaken="1" maxpossiblescore="0"
      asset-id="2006334911" />
    <row num-questions="2" average-score="13" low-score="0" high-score="20"
      numtaken="3" numdistincttaken="3" maxpossiblescore="20"
      asset-id="2006334911" />
  </report-quiz-summary>
</results>
```

**report-quiz-takers****Availability**

Breeze 4

**Description**

Provides information about all users who have taken a quiz in a training. Use a `sco-id` to identify the quiz.

To reduce the volume of the response, use any allowed filter or pass a `type` parameter to return information about just one type of SCO (courses, presentations, or meetings).

**Request URL**

```
http://server_name/api/xml
  ?action=report-quiz-takers
  &sco-id=integer
  &principal-id=integer
  &type=allowedValue
  &filter-definition=value
  &sort-definition=value
  &session=BreezeSessionCookieValue
```

**Action reference**

**Parameters**

Name	Type	Required	Description
sco-id	Integer	Y	The unique ID of a presentation or course that contains a quiz.
principal-id	Integer	N	The ID of a principal for whom you want quiz results.
type	Allowed value	N	The type of content for which you want results. Allowed values are <code>course</code> , <code>presentation</code> , and <code>meeting</code> .
<a href="#">filter-definition</a>	Filter definition	N	A filter to reduce the volume of the response.
<a href="#">sort-definition</a>	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the <code>BREEZESSESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-quiz-takers>
    <row transcript-id=integer sco-id=integer principal-id=integer
      status=allowedValue score=integer asset-id=integer
      permission-id=allowedValue attempts=integer time-taken=integer
      certificate=integer answered-survey=boolean version=integer>
      <name>string</name>
      <login>string</login>
      <date-created>datetime</date-created>
      <principal-name>string</principal-name>
      <override>boolean</override>
    </row>
  </report-quiz-takers>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-quiz-takers		Container	Information about all users who have taken the quiz.
row		Container	Information about one user who has taken the quiz.
	transcript-id	Integer	The ID of the transcript on which the user's quiz score is recorded.
	sco-id	Integer	The unique ID of the presentation, course, or meeting that has the quiz.
	principal-id	Integer	The ID of the user who took the quiz.

**Action reference**

Element	Attribute	Type	Description
	status	Allowed value	Whether the user passed or failed the most recent attempt at the quiz. Allowed values are <code>user-passed</code> and <code>user-failed</code> .
	score	Integer	The user's score on the most recent attempt at the quiz.
	asset-id	Integer	The ID of the version of the quiz the user attempted.
	permission-id	Allowed value	The level of permission the user has to access the quiz (see <a href="#">permission-id</a> for values).
	attempts	Integer	The number of times the user has taken the quiz.
	time-taken	Integer	The amount of time the user spent taking the quiz, in milliseconds.
	certificate	Integer	The unique ID of a user's transcript.
	answered-survey	Boolean	Whether the learner completed a quiz. If 0 or false, the training does not have a quiz or the learner did not complete it. If 1 or true, the learner completed the quiz.
	version	Integer	The revision number of the quiz.
name		String	The name of the quiz.
login		String	The user's login name on the server.
date-created		Datetime	The date and time of the user's most recent quiz attempt.
principal-name		String	The full name of the user taking the quiz.
override		Boolean	A setting indicating whether a training manager can change the user's score on the quiz.

**Sample request**

`https://example.com/api/xml?action=report-quiz-takers&sco-id=2006334909`

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-quiz-takers>
    <row transcript-id="2006337854" sco-id="2006334909"
      principal-id="2006258745" status="incomplete" score="0"
      max-score="20" asset-id="2006334911" permission-id=""
      attempts="4" time-taken="12593" certificate="" answered-survey="1"
      version="1">
      <name>California State Quiz</name>
      <login>joy@acme.com</login>
      <date-created>2006-05-16T11:14:47.000-07:00</date-created>
      <principal-name>Joy Smith</principal-name>
      <override>>false</override>
    </row>
  </report-quiz-takers>
</results>
```

## report-quotas

### Availability

Breeze 4

### Description

Returns information about the quotas that apply to your Adobe Connect license or Adobe Connect hosted account. Adobe Connect enforces various quotas, for example, the number of concurrent users in training, the number of downloads, the number of authors, and so on.

Although your server license determines certain quotas, you can scale your license beyond your limit. In the response from `report-quotas`, the `soft-limit` is the number defined by your license. The `soft-limit` is the same as the `limit`, unless you purchase a Burst Pack for meetings, which allows additional participants to join past the limit, on an overage basis.

### Request URL

```
http://server_name/api/xml
  ?action=report-quotas
  &session=BreezeSessionCookieValue
```

### Parameters

Name	Type	Required	Description
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

### Filters

Results cannot be filtered or sorted.

### Response structure

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-quotas>
    <quota acl-id=integer quota-id=string used=integer
      limit=allowedValue soft-limit=integer>
      <date-begin>datetime</date-begin>
      <date-end>datetime</date-end>
    </quota>
  </report-quotas>
</results>
```

### Response values

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Action reference**

Element	Attribute	Type	Description
report-quotas		Container	Information about all of the quotas set for the account.
quota		Container	Information about one quota.
	acl-id	Integer	The ID of the account on which the quota is defined.
	quota-id	Allowed value	The name of the quota defined by the server, ending in -quota.
	used	Integer	The number of uses that count toward this quota.
	limit	Integer	The limit at which the server does not allow access. Has the same value as soft-limit, unless you have purchased a Burst Pack to allow for overage. The value is either an integer or unlimited.
	soft-limit	Integer	The limit determined by your server license.
date-begin		Datetime	The date and time the quota was effective on the server.
date-end		Datetime	The date the quota expires.

**Sample request**

`https://example.com/api/xml?action=report-quotas`

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-quotas>
    <quota acl-id="624520" quota-id="download-quota"
      used="1" limit="unlimited" soft-limit="1000000000">
      <date-begin>2004-03-09T09:45:41.047-08:00</date-begin>
      <date-end>3000-01-01T00:00:00.000-08:00</date-end>
    </quota>
    <quota acl-id="624520" quota-id="bandwidth-quota" used="12802"
      limit="unlimited" soft-limit="1000000000">
      <date-begin>2006-05-31T17:00:00.943-07:00</date-begin>
      <date-end>2006-06-30T17:00:00.943-07:00</date-end>
    </quota>
    ...
  </report-quotas>
</results>
```

**report-sco-slides**

**Availability**

Breeze 4

**Description**

Returns information about the slides in a presentation. The information includes how many times, and how recently, each slide has been viewed.

**Action reference**

**Request URL**

```
http://server_name/api/xml
    ?action=report-sco-slides
    &sco-id=integer
    &asset-id=integer
    &sort-definition=value
    &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
sco-id	Integer	Y	The unique ID of a presentation.
asset-id	Integer	N	The version number of a presentation, incremented each time a presentation is uploaded.
<a href="#">sort-definition</a>	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the BREEZESESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can sort the response on any element or attribute it contains.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-sco-slides>
    <row slide=integer name=integer asset-id=integer views=integer>
      <date-created>datetime</date-created>
    </row>
    ...
  </report-sco-slides>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-sco-slides		Container	Information about all of the slides in a presentation, indicating how many times and how recently a slide has been viewed.
row		Container	Information about one slide in the presentation.
	slide	Integer	The number of the slide within the presentation.
	name	Integer	The name of the slide in the presentation.



**Action reference**

Element	Attribute	Type	Description
	asset-id	Integer	The version number of the presentation. Each time a presentation is published, it has a new <code>asset-id</code> .
	views	Integer	The number of times the slide has been viewed.
date-created		Datetime	The date and time the slide was last viewed.

**Sample request**

`https://example.com/api/xml?action=report-sco-slides&sco-id=2006334909`

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-sco-slides>
    <row slide="1" name="1" views="4">
      <date-created>2006-05-16T11:14:54.453-07:00</date-created>
    </row>
    <row slide="2" name="2" views="4">
      <date-created>2006-05-16T11:14:59.593-07:00</date-created>
    </row>
    <row slide="3" name="3" views="3">
      <date-created>2006-05-12T11:55:52.330-07:00</date-created>
    </row>
    <row slide="4" name="4" views="3">
      <date-created>2006-05-12T11:55:55.487-07:00</date-created>
    </row>
    <row slide="5" name="5" views="3">
      <date-created>2006-05-12T11:56:00.233-07:00</date-created>
    </row>
  </report-sco-slides>
</results>
```

**See also**

[report-sco-views](#)

**report-sco-views****Availability**

Breeze 4

**Description**

Indicates how many times, and how recently, a SCO was viewed.

**Request URL**

```
http://server_name/api/xml
  ?action=report-sco-views
  &sco-id=integer
  &session=BreezeSessionCookieValue
```

**Action reference**

**Parameters**

Name	Type	Required	Description
sco-id	Integer	Y	The unique ID of a SCO to check for views.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-sco-views sco-id=integer type=allowedValue is-folder=boolean
    views=integer>
    <name>string</name>
    <last-viewed-date>string</last-viewed-date>
  </report-sco-views>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-sco-views		Container	Information about how many times, and how recently, the presentation was viewed.
	sco-id	Integer	The unique ID of the presentation.
	type	Allowed value	The type of content object (SCO). Allowed values are <code>content</code> , <code>curriculum</code> , <code>event</code> , <code>folder</code> , <code>link</code> , <code>meeting</code> , and <code>tree</code> .
	is-folder	Boolean	A value indicating whether the SCO is a folder (if 1) or another type of object (if 0).
	views	Integer	The number of times users have viewed the SCO.
name		String	The name of the SCO.
last-viewed-date		Datetime	The date and time the SCO was last viewed.

**Sample request**

<https://server.com/api/xml?action=report-sco-views&sco-id=2006334909>

**Action reference****Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-sco-views sco-id="2006334909" type="content" is-folder="0"
    views="3">
    <name>Quiz on California</name>
    <last-viewed-date>2006-05-12T11:55:24.940-07:00</last-viewed-date>
  </report-sco-views>
</results>
```

**report-user-trainings-taken****Availability**

Connect Enterprise 6

**Description**

Returns a list of all courses and curriculums a user has taken, whether or not the user has completed the training. Each course or curriculum is returned in a separate `row` element and has the most recent transcript of the user's scores.

**Request URL**

```
http://server_name/api/xml
  ?action=report-user-trainings-taken
  &principal-id=integer
  &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
principal-id	Integer	Y	The ID of a user for whom you want a list of trainings.
session	String	N	The value of the BREEZESESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

**Action reference**

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-user-trainings-taken>
    <row transcript-id=integer max-retries=integer sco-id=integer
      type=allowedValue icon=allowedValue status=allowedValue
      certificate=integer score=integer permission-id=allowedValue
      attempts=allowedValue>
      <name>string</name>
      <description>string</description>
      <url-path>string</url-path>
      <date-taken>datetime</date-taken>
      <from-curriculum>boolean</from-curriculum>
    </row>
    ...
  </report-user-trainings-taken>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
	type	Allowed value	The type of the SCO. Allowed values are <code>user-content</code> , <code>content</code> , and <code>my-content</code> .
report-user-trainings-taken		Container	A list of all trainings the user has attempted.
row		Container	Information about one course or curriculum the user has taken, whether or not passed.
	transcript-id	Integer	The ID of the record of the user's most recent score on this training.
	max-retries	Integer	The maximum number of times the user can repeat the training.
	sco-id	Integer	The ID of the training SCO.
	type	Allowed value	The type of the training SCO (see allowed values at <a href="#">type</a> ).
	icon	Allowed value	The type of icon that identifies the course or curriculum in Connect Central. Provides information about the course or curriculum in addition to its <code>type</code> (see allowed values at <a href="#">icon</a> ).
	status	Allowed value	The status of the user's work with the SCO. Allowed values for a course or presentation are <code>user-passed</code> , <code>user-failed</code> , <code>completed</code> , <code>incomplete</code> , <code>not-attempted</code> , and <code>review</code> . A curriculum or folder can only be <code>completed</code> or <code>incomplete</code> .
	certificate	Integer	The ID of the record that shows the user passed or completed the training.

**Action reference**

Element	Attribute	Type	Description
	score	Integer	The score the user earned on the most recent attempt at the training.
	permission-id	Allowed value	The permission the user has been assigned to access the course or curriculum (see <a href="#">permission-id</a> for allowed values).
	attempts	Integer	The number of attempts the user has made at this training.
name		String	The name of the training SCO.
description		String	The description of the training SCO.
url-path		String	The unique identifier of the SCO that appears in its URL after the domain name.
date-taken		Datetime	The date the user interacted with the training SCO (viewed a presentation, took a quiz, and so on).
from-curriculum		Boolean	A value indicating whether this course was taken as part of a curriculum.

**Sample request**

```
https://example.com/api/xml?action=report-user-trainings-taken
&principal-id=2006258745&principal-id=4797406
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-user-trainings-taken>
    <row transcript-id="2006745722" max-retries="" sco-id="2006745673"
      type="content" icon="course" status="user-passed"
      certificate="2006745722" score="0" permission-id=""
      attempts="1">
      <name>All About Web Communities</name>
      <description>test</description>
      <url-path>/p33096345/</url-path>
      <date-taken>2006-06-12T15:06:02.947-07:00</date-taken>
      <from-curriculum>>false</from-curriculum>
    </row>
    ...
  </report-user-trainings-taken>
</results>
```

**report-user-training-transcripts****Availability**

Connect Enterprise 6

**Description**

Returns a list of transcripts for trainings a user has taken. A *transcript* is the record of one score a user obtained from one attempt at taking one training. A training can be a course, curriculum, meeting, or event.

**Action reference**

The response can include more than one transcript for a training SCO, if the user has attempted the training more than once. A user can fail a training the first time and then pass on the second attempt. Each attempt has its own transcript, and both transcripts are included in the report.

**Request URL**

```
http://server_name/api/xml
    ?action=report-user-training-transcripts
    &principal-id=integer
    &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
principal-id	Integer	Y	The ID of a user whose transcripts you want.
session	String	N	The value of the BREEZESESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

You can filter or sort the response on any element or attribute it contains.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <report-user-training-transcripts>
    <row transcript-id=integer sco-id=integer
      principal-id=integer status=allowedValue score=integer
      max-score=integer certificate=integer type=allowedValue
      icon=allowedValue>
      <name>string</name>
      <url-path>string</url-path>
      <login>string</login>
      <date-taken>datetime</date-taken>
      <principal-name>string</principal-name>
      <sco-tag>string</sco-tag>
    </row>
    ...
  </report-user-training-transcripts>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
report-user-training-transcripts		Container	Information about all transcripts for the specified user.

**Action reference**

Element	Attribute	Type	Description
row		Container	Information about one attempt the user made on the training.
	transcript-id	Integer	The ID of the transcript on which the user's score is recorded. A distinct transcript exists for each of the user's attempts at taking the SCO.
	sco-id	Integer	The ID of the SCO, which can be a meeting, content, course, curriculum, event, or seminar.
	principal-id	Integer	The ID of the user.
	status	Allowed value	The status of the user's work with the SCO (see <a href="#">status attribute</a> for allowed values).
	score	Integer	The score the user earned on the SCO. If the SCO does not have a score, as with a meeting, the value of <code>score</code> is 0.
	max-score	Integer	The maximum score possible on the course or curriculum.
	certificate	Integer	The ID of the record of the courses and curriculums the user has passed or completed.
	type	Allowed value	The type of the SCO. Allowed values are <code>user-content</code> , <code>content</code> , and <code>my-content</code> .
	icon	Allowed value	The name of the icon that identifies the course or curriculum in Adobe Connect Central.
name		String	The name of the course or curriculum.
url-path		String	The unique identifier of the course or curriculum that appears in its URL after the domain name.
login		String	The user's login ID on Adobe Connect Server.
date-taken		Datetime	The date the user interacted with the course or curriculum (viewed a presentation, took a quiz, and so on).
principal-name		String	The name of the user interacting with the SCO.
sco-tag		String	A descriptive label for the SCO, in addition to the name (for example, a short course name).

**Sample request**

```
https://example.com/api/xml?action=report-user-training-transcripts
&principal-id=2006258745
```

**Action reference****Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-user-training-transcripts>
    <row transcript-id="2006905612" sco-id="2006298431"
      principal-id="2006258745" status="review" score="0" max-score=""
      certificate="" type="content" icon="course">
      <name>Test Course</name>
      <url-path>/test/</url-path>
      <login>joy@acme.com</login>
      <date-taken>2006-06-30T15:23:55.070-07:00</date-taken>
      <principal-name>Joy Smith</principal-name>
    </row>
    <row transcript-id="2007016805" sco-id="2006298431"
      principal-id="2006258745" status="review" score="0" max-score=""
      certificate="" type="content" icon="course">
      <name>Test Course</name>
      <url-path>/test/</url-path>
      <login>joy@acme.com</login>
      <date-taken>2006-07-14T16:55:28.440-07:00</date-taken>
      <principal-name>Joy Smith</principal-name>
    </row>
  </report-user-training-transcripts>
</results>
```

**sco-by-url****Availability**

Connect Pro 7

**Description**

Returns information about a SCO at a specified URL path. The URL path is the unique identifier after the domain name in the URL to the SCO. For example, if you have a meeting with the custom URL

<http://example.acrobat.com/teammeeting>, the URL path is `/teammeeting`. If you pass the full URL path, Connect returns the status code "no data".

**Request URL**

```
http://server_name/api/xml
  ?action=sco-by-url
  &url-path=url
```

**Parameters**

Name	Required	Description
url-path	Y	The unique identifier after the domain name in the URL to the SCO.

**Filters**

You cannot filter or sort the response.



**Action reference**

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <owner-principal type="allowedValue" principal-id="number" account-id="number" has-children="Boolean" is-hidden="Boolean" is-primary="Boolean">
    <ext-login></ext-login>
    <login></login>
    <name></name>
    <email></email>
  </owner-principal>
  - <sco sco-id="number" account-id="number" display-seq="number" folder-id="number" icon="string" lang="string" max-retries="number" source-sco-id="number" type="allowedValue" version="number">
    <url-path></url-path>
    <date-begin></date-begin>
    <date-created></date-created>
    <date-modified></date-modified>
    <name></name>
  </sco>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
owner-principal		Container	Contains information about a principal.
	type	Allowed value	The type of principal (see <a href="#">type</a> for values).
	principal-id	Integer	The ID of the principal.
	account-id	Integer	The ID of the account the principal belongs to.
	has-children	Boolean	Whether the principal has children. Groups have children and users don't, so this attribute indicates whether the principal is a group.
	is-hidden	Boolean	Whether the principal is hidden ( <code>true</code> ) or not ( <code>false</code> ) in Connect Central or your application.
	is-primary	Boolean	Whether the principal is a built-in group ( <code>true</code> ) or not ( <code>false</code> ).
ext-login		String	The login ID sent from an external network. By default, the same value as login.
login		String	The login name of the user who is logged in to the server, often the user's e-mail address.
name		String	The name of the user who is logged in to the server.
e-mail		String	The e-mail address of the user who is logged in to the server.
sco		Container	One object within the folder.
	sco-id	Integer	The unique ID of an object.

**Action reference**

Element	Attribute	Type	Description
	account-id	Integer	The unique ID of an account.
	display-seq	Integer	The sequence in which Connect Central or your application displays the field.
	folder-id	Integer	The ID of a folder.
	icon	Allowed value	The name of the icon that visually identifies this object.
	lang	String	The language in which information about the SCO is displayed (see <a href="#">lang</a> for values).
	max-retries	Integer	The maximum number of times a user can retake the course or curriculum. If a user can take the course 3 times, <code>max-retries</code> is 2.
	source-sco-id	Integer	The unique ID of a content SCO used in a course or curriculum.
	type	Allowed value	The type of the object (see <a href="#">type</a> for values).
	version	Integer	A sequential ID for the session, starting at 1.
url-path		String	The unique identifier after the domain name in the URL to the SCO.
date-begin		Datetime	The beginning date of a course or meeting (returned for a course or meeting only).
date-created		Datetime	The date a course or meeting was created (returned for a course or meeting only).
date-modified		Datetime	The date the object was last modified.
name		String	The name of the object on the server.

**Sample request**

```
http://example.acrobat.com/api/xml?action=sco-by-url&url-path=/p18656190/
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <owner-principal type="content" principal-id="824622254" account-id="824592506" has-
children="false" is-hidden="false" is-primary="false">
    <ext-login>jdoe@adobe.com</ext-login>
    <login>jdoe@adobe.com</login>
    <name>Jane Doe</name>
    <email>jdoe@adobe.com</email>
  </owner-principal>
  - <sco sco-id="825344405" account-id="824592506" display-seq="0" folder-id="824622258"
icon="curriculum" lang="en" max-retries="" source-sco-id="" type="curriculum" version="0">
    <url-path>/p18656190</url-path>
    <date-begin>2009-09-15T13:30:00.000-07:00</date-begin>
    <date-created>2009-09-15T13:32:45.683-07:00</date-created>
    <date-modified>2009-09-15T13:32:45.683-07:00</date-modified>
    <name>Test Curriculum</name>
  </sco>
</results>
```

## sco-contents

### Availability

Breeze 4

### Description

Returns a list of SCOs within another SCO. The enclosing SCO can be a folder, meeting, or curriculum.

In general, the contained SCOs can be of any type—meetings, courses, curriculums, content, events, folders, trees, or links (see the list in [type](#)). However, the type of the contained SCO needs to be valid for the enclosing SCO. For example, courses are contained within curriculums, and meeting content is contained within meetings.

Because folders are SCOs, the returned list includes SCOs and subfolders at the next hierarchical level, but not the contents of the subfolders. To include the subfolder contents, call [sco-expanded-contents](#).

### Request URL

```
http://server_name/api/xml
  ?action=sco-contents
  &sco-id=integer
  &filter-definition=value
  &sort-definition=value
  &session=value
```

### Parameters

Name	Type	Required	Description
sco-id	Integer	Y	The unique ID of a folder for which you want to list contents. You can get the <code>sco-id</code> by calling <code>sco-shortcuts</code> .
<a href="#">filter-definition</a>	Filter definition	N	A filter to reduce the volume of the response.
<a href="#">sort-definition</a>	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the <code>BREEZESSESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

### Filters

You can filter the response on any element or attribute, with these exceptions:

- You cannot filter on `duration`.
- If you use `filter-date-begin`, `filter-date-end`, or `filter-date-modified`, specify a time without a time zone, for example:

```
filter-date-modified=2005-01-05T10:44:03
```

You can use `filter-gt` or `filter-lt` with a date field and a full date, including the time zone.

You can sort the response on any element or attribute.

**Action reference**

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <scos>
    <sco sco-id=integer source-sco-id=integer folder-id=integer
      type=allowedValue icon=allowedValue display-seq=integer
      is-folder=boolean byte-count=integer ref-count=integer>
      <name>string</name>
      <url-path>string</url-path>
      <description>string</description>
      <date-begin>string</date-begin>
      <date-modified>datetime</date-modified>
      <date-end>string</date-end>
      <sco-tag>string</sco-tag>
    </sco>
  </scos>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
scos		Container	The list of objects within the folder.
sco		Container	One object within the folder.
	sco-id	Integer	The unique ID of one object within the folder.
	source-sco-id	Integer	The unique ID of a content SCO used in a course or curriculum.
	folder-id	Integer	The ID of a folder, passed as <code>sco-id</code> in the request.
	type	Allowed value	The type of the object (see <a href="#">type</a> for values). SCOs that represent content have a <code>type</code> of <code>content</code> , rather than a more specific type, such as <code>presentation</code> .
	icon	Allowed value	The name of the icon that identifies the object. Provides more detail on the <code>type</code> of object in <code>type</code> .
	display-seq	Integer	The sequence in which Connect Central or your application displays the object.
	is-folder	Boolean Integer	A value indicating whether the object is a folder (1) or not (0).
	byte-count	Integer	The size of the content. For folders, this value will be 0.
	ref-count	Integer	The number of SCOs that reference this SCO.
name		String	The name of the object on the server.
url-path		String	The unique identifier after the domain name in the URL to the SCO.
description		String	The description of the object.
date-modified		Datetime	The date the object was last modified.

**Action reference**

Element	Attribute	Type	Description
date-begin		Datetime	The beginning date of a course or meeting (returned for a course or meeting only).
date-end		Datetime	The end date of a course or meeting (returned for a course or meeting only).
domain-name		String	The domain name at which you can access a meeting or event (returned for meetings and events only).
duration		Datetime	The length of time a course or meeting lasted (returned for a course or meeting only).
sco-tag		String	A brief description of the SCO.

**Sample request**

```
https://example.com/api/xml?action=sco-contents&sco-id=2006258748
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <scos>
    <sco sco-id="2007035246" source-sco-id="2006334909"
      folder-id="2006258748" type="content" icon="course"
      display-seq="0" is-folder="0">
      <name>Java 101</name>
      <url-path>/java101</url-path>
      <date-begin>2006-07-20T17:15:00.000-07:00</date-begin>
      <date-modified>2006-07-20T17:21:38.860-07:00</date-modified>
    </sco>
  </scos>
</results>
```

**See also**

[sco-expanded-contents](#), [sco-shortcuts](#)

**sco-delete****Availability**

Breeze 4

**Description**

Deletes one or more objects (SCOs).

If the `sco-id` you specify is for a folder, all the contents of the specified folder are deleted. To delete multiple SCOs, specify multiple `sco-id` parameters.

You can use a call such as `sco-contents` to check the `ref-count` of the SCO, which is the number of other SCOs that reference this SCO. If the SCO has no references, you can safely remove it, and the server reclaims the space.

If the SCO has references, removing it can cause the SCOs that reference it to stop working, or the server not to reclaim the space, or both. For example, if a course references a quiz presentation, removing the presentation might make the course stop working.

**Action reference**

As another example, if a meeting has used a content SCO (such as a presentation or video), there is a reference from the meeting to the SCO. Deleting the content SCO does not free disk space, because the meeting still references it.

To delete a SCO, you need at least `manage` permission (see [permission-id](#) for details). Users who belong to the built-in authors group have `manage` permission on their own content folder, so they can delete content within it.

**Request URL**

```
http://server_name/api/xml
  ?action=sco-delete
  &sco-id=integer
  &session=value
```

**Parameters**

Name	Type	Required	Description
sco-id	Integer	Y	The unique ID of a SCO.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Sample request**

```
https://example.com/api/xml?action=sco-delete&sco-id=2007171127
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
</results>
```

**See also**

[sco-info](#), [sco-move](#), [sco-nav](#), [sco-expanded-contents](#)

## sco-expanded-contents

### Availability

Breeze 5

To list the contents of a curriculum in Connect Pro 7 and later, use `curriculum-contents`.

### Description

Lists all of the SCOs in a folder, including the contents of subfolders, curriculums, and any type of enclosing SCO.

**Note:** If you call this command on a large folder—such as the root meeting folder for a large account—the amount of data returned is very large.

If you do not use a filter, the list of SCOs is returned in the same order as it appears in Connect Central. If you use a filter or a sort, the list is returned according to the filter or sort you use.

### Request URL

```
http://server_name/api/xml
?action=sco-expanded-contents
&sco-id=integer
&filter-definition=value
&sort-definition=value
&session=BreezeSessionCookieValue
```

### Parameters

Name	Type	Required	Description
sco-id	Integer	Y	The unique ID of a folder.
<code>filter-definition</code>	Filter definition	N	A filter to reduce the volume of the response.
<code>sort-definition</code>	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the <code>BREEZESSESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

### Filters

You can filter the response on any element or attribute, with these exceptions:

- You cannot filter on `duration`.
- If you use `filter-date-begin`, `filter-date-end`, or `filter-date-modified`, specify a date in ISO 8601 format but without a time zone, for example:

```
filter-date-modified=2005-01-05T10:44:03
```

However, you can use `filter-gt-datefield` or `filter-lt-datefield` with a full date that includes a time zone.

- Do not use partial match filters constructed with `filter-like` (such as `filter-like-name`), as they might affect server performance.

You can sort the response on any element or attribute except `date-begin`, `date-created`, `date-modified`, and `url-path`.

**Action reference**

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <expanded-scoss>
    <sco depth=integer sco-id=integer folder-id=integer type=allowedValue
      icon=allowedValue lang=allowedValue source-sco-id=integer
      display-seq=integer source-sco-type=integer>
      <name>string</name>
      <url-path>string</url-path>
      <date-created>datetime</date-created>
      <date-modified>datetime</date-modified>
    </sco>
    ... more sco elements ...
  </expanded-scoss>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
expanded-scoss		Container	The list of all SCOs the folder contains.
sco		Container	Details about one SCO in the folder. This SCO can be a folder or any other type of object.
	depth	Integer	The depth in the content tree at which this object appears, with top-level objects at 1.
	sco-id	Integer	The unique ID of the SCO. If the SCO is a folder, same as <code>folder-id</code> .
	folder-id	Integer	The ID of the folder the SCO belongs to.
	type	Allowed value	The type of this content object (see <a href="#">type</a> ).
	icon	Allowed value	The name of the icon that visually identifies this object.
	lang	Allowed value	The language in which information about the SCO is displayed (see <a href="#">lang</a> for values).
	source-sco-id	Integer	The ID of a SCO from which this SCO was created, such as a meeting template or course content.
	display-seq	Integer	The sequence in which Connect Central (or your application, if you use this value) displays a list of SCOs. Values are not necessarily unique, so multiple SCOs can have the same <code>display-seq</code> value. In that case, the application must define the display sequence. The default is 0.
	source-sco-type	Integer	An integer indicating the type of the SCO from which this SCO was created.

**Sample request**

```
https://example.com/api/xml?action=sco-expanded-contents&sco-id=624529
```



**Action reference**

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <expanded-scops>
    <sco depth="0" sco-id="624529" folder-id="624520" type="folder"
      icon="folder" lang="en" source-sco-id="" display-seq="0"
      source-sco-type="">
      <name>Shared Meetings</name>
      <url-path>/f624529/</url-path>
      <date-created>2004-03-09T09:45:41.060-08:00</date-created>
      <date-modified>2005-03-18T10:19:38.950-08:00</date-modified>
    </sco>
    <sco depth="1" sco-id="2598379" folder-id="624529" type="meeting"
      icon="meeting" lang="en" source-sco-id="-8888" display-seq="0"
      source-sco-type="3">
      <name>Monday Night Football</name>
      <url-path>/r68075204/</url-path>
      <description>Monday Night Football</description>
      <date-begin>2004-05-17T15:30:00.000-07:00</date-begin>
      <date-end>2004-05-18T00:15:00.000-07:00</date-end>
      <date-created>2004-05-17T15:50:39.733-07:00</date-created>
      <date-modified>2006-08-16T00:34:52.930-07:00</date-modified>
    </sco>
  </expanded-scops>
</results>
```

**sco-info**

**Availability**

Breeze 4

**Description**

Provides information about a SCO on Adobe Connect. The object can have any valid SCO type. See [type](#) for a list of the allowed SCO types.

The response includes the account the SCO belongs to, the dates it was created and last modified, the owner, the URL that reaches it, and other data. For some types of SCOs, the response also includes information about a template from which this SCO was created.

**Request URL**

```
http://server_name/api/xml
  ?action=sco-info
  &sco-id=integer
  &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
sco-id	Integer	Y	The unique ID of a SCO on the server.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Action reference**

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <sco account-id=integer disabled=datetime display-seq=integer
    folder-id=integer icon=allowedValue lang=allowedValue
    max-retries=integer sco-id=integer source-sco-id=integer
    type=allowedValue version=integer>
    <date-begin>datetime</date-begin>
    <date-created>datetime</date-created>
    <date-end>datetime</date-end>
    <date-modified>datetime</date-modified>
    <description>string</description>
    <name>string</name>
    <url-path>string</url-path>
    <passing-score>integer</passing-score>
    <duration>datetime</duration>
    <section-count>integer</section-count>
  </sco>
  <source-sco>
    <source-sco account-id=integer display-seq=integer folder-id=integer
      icon=allowedValue lang=allowedValue max-retries=integer
      sco-id=integer source-sco-id=integer type=allowedValue
      version=integer>
    <date-created>datetime</date-created>
    <date-modified>datetime</date-modified>
    <name>string</name>
    <url-path>string</url-path>
  </source-sco>
</source-sco>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
sco		Container	Information about the SCO.
	account-id	Integer	The ID of the account the SCO belongs to.
	disabled	Datetime	An empty value if the SCO has not been disabled. If it has, the date and time it was disabled.
	display-seq	Integer	The sequence in which Connect Central (or your application, if you use this value) displays a list of SCOs. Values are not necessarily unique, so multiple SCOs can have the same <code>display-seq</code> value. In that case, the application must define the display sequence. The default is 0.

Action reference

Element	Attribute	Type	Description
	folder-id	Integer	The ID of the folder the SCO belongs to.
	icon	Allowed value	The type of icon used as a visual identifier for the SCO (see <a href="#">icon</a> ).
	lang	Allowed value	An abbreviation for the new language (see <a href="#">type</a> for values).
	max-retries	Integer	The number of times the user is allowed to attempt to take the SCO.
	sco-id	Integer	The unique ID of the SCO.
	source-sco-id	Integer	The unique ID of a template from which the SCO is derived.
	type	Allowed value	The content type of the SCO (see <a href="#">type</a> for values). <code>type</code> is a high-level category. <code>icon</code> provides more detail on the type of content.
	version	Integer	The version number of the SCO, incremented when the object is modified or uploaded to the server.
date-begin		Datetime	If the SCO is a meeting, the date and time the meeting starts.
date-created		Datetime	The date and time the SCO was created (or, for content, uploaded).
date-end		Datetime	If the SCO is a meeting, the date and time the meeting ends.
date-modified		Datetime	The date and time the SCO was last modified.
description		String	The description of the SCO entered when the SCO was created.
name		String	The name of the SCO.
url-path		String	The path to the SCO on the server.
passing-score		Integer	The minimum score that a user must have to pass a training course.
duration		Integer	The length of time needed to view or play the SCO, in milliseconds.
section-count		Integer	The number of sections in the course content, including the number of slides, pages, chapters, interactions, or other content divisions.
source-sco		Container	Information about any SCOs that are templates for, or provide content to, the SCO you are interested in. The SCOs that can have a source are meetings, courses, or events.
source-sco		Container	Details about one SCO that is a template for, or provides source content to, the SCO you are interested in. Has additional elements and attributes, the same as the <code>sco</code> element.

Sample request

<https://example.com/api/xml?action=sco-info&sco-id=2006320683>

**Action reference****Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
  <results>
    <status code="ok" />
    <sco account-id="624520" disabled="" display-seq="0"
      folder-id="2006258750" icon="meeting" lang="en" max-retries=""
      sco-id="2006320683" source-sco-id="-1625529" type="meeting"
      version="0">
      <date-begin>2006-05-04T11:15:00.000-07:00</date-begin>
      <date-created>2006-05-04T11:27:47.087-07:00</date-created>
      <date-end>2006-05-04T12:15:00.000-07:00</date-end>
      <date-modified>2006-05-04T11:27:47.087-07:00</date-modified>
      <name>Technology and Law Review Meeting</name>
      <url-path>/tlawreview/</url-path>
    </sco>
    <source-sco>
      <source-sco account-id="624520" display-seq="0" folder-id="-625529"
        icon="meeting" lang="en" max-retries="" sco-id="-1625529"
        source-sco-id="-8888" type="meeting" version="0">
        <date-created>2004-10-05T00:49:30.217-07:00</date-created>
        <date-modified>2005-01-04T15:03:25.937-08:00</date-modified>
        <name>Default Meeting Template</name>
        <url-path>/defaultMeetingTemplate/</url-path>
      </source-sco>
    </source-sco>
  </results>
```

**sco-move****Availability**

Breeze 4

**Description**

Moves a SCO from one folder to another.

To move a SCO to a folder, the current user must have permission to create content in the target folder. In general, users have permission on their own folders (such as `my-meetings`, `my-courses`, `my-events`, `my-content`, and `my-meeting-templates`) by default. To move SCOs to a shared folder such as `content`, `courses`, and `meetings`, a user must have `Manage` permission or be an Administrator.

**Request URL**

```
http://server_name/api/xml
  ?action=sco-move
  &folder-id=integer
  &sco-id=integer
  &session=BreezeSessionCookieValue
```

**Action reference****Parameters**

Name	Type	Required	Description
folder-id	Integer	Y	The ID of the destination folder.
sco-id	Integer	Y	The unique ID of the SCO to move.
session	String	N	The value of the BREEZESESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Sample request**

```
https://example.com/api/xml?action=sco-move&sco-id=2006744233
&folder-id=2006258748
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
<status code="ok" />
</results>
```

**See also**

[sco-nav](#)

**sco-nav****Availability**

Breeze 4

**Description**

Describes the folder hierarchy that contains a SCO.

**Action reference**

The `sco-nav` call is useful for creating a navigation tree, breadcrumb trail, or any other type of user interface hierarchy. The response contains a list of `sco` elements, one for the SCO you are querying and one for each of its enclosing folders up to the top-level folder. The top-level folder is one of the list of folders returned by `sco-shortcuts`.

In each `sco` element, the `depth` attribute indicates how many hierarchical levels the SCO is from the SCO you specify in the request. A `depth` of 0 indicates the SCO you are querying, a `depth` of 1 indicates the folder that contains the SCO, and so on.

**Request URL**

```
http://server_name/api/xml
    ?action=sco-nav
    &sco-id=integer
    &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
sco-id	Integer	Y	The unique ID of a SCO for which you want a folder hierarchy up to the root level.
session	String	N	The value of the <code>BREEZESSESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <sco-nav>
    <sco sco-id=integer type=allowedValue icon=allowedValue depth=integer>
      <name>string</name>
    </sco>
    ...
  </sco-nav>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <code>status</code> ).
sco-nav		Container	The entire navigation tree from the top-level folder to the SCO.
sco		Container	Information about one SCO in the hierarchy.
	sco-id	Integer	The unique ID of the SCO.
	type	Allowed value	The type of SCO (see <code>type</code> for values).

**Action reference**

Element	Attribute	Type	Description
	icon	Allowed value	The icon that visually represents the SCO (see <a href="#">icon</a> for values).
	depth	Integer	A number representing the level of a SCO in the folder hierarchy relative to the SCO passed in the request (0 for the passed SCO, 1 for one level above, and so on). Values increase as you move up the hierarchy toward the top-level folder.
name		String	The name of the SCO.

**Sample request**

`https://example.com/api/xml?action=sco-nav&sco-id=2006334909`

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <sco-nav>
    <sco sco-id="624522" type="folder" icon="folder" depth="2">
      <name>User Content</name>
    </sco>
    <sco sco-id="2006258747" type="folder" icon="folder" depth="1">
      <name>joy@acme.com</name>
    </sco>
    <sco sco-id="2006334909" type="content" icon="producer" depth="0">
      <name>Test Quiz</name>
    </sco>
  </sco-nav>
</results>
```

**See also**

[sco-move](#)

**sco-search**

**Availability**

Breeze 4

**Description**

Provides a list of all SCOs that have content matching the search text.

The `sco-search` action searches the content of some types of SCOs for the query string. The types of SCOs searched include presentation archives, meeting archives, and the presentation components of a course or curriculum. A presentation that is included in a course returns two sets of results, one for the actual presentation and one for the course. The search does not include the SCO name or any metadata about the SCO stored in the database.

The query is not case-sensitive and allows wildcards at the end of a query string. The allowed wildcards are:

- An asterisk (\*) to match any character or characters
- A question mark (?) to match any one character

For example, you can use the query strings `quiz`, `qu*`, or `qui?`. However, you cannot use a wildcard at the beginning or within a query string.

You can also use the operators `and` and `or` to return multiple matches, with spaces separating the operator and the search terms, like this:

```
https://example.com/api/xml?action=sco-search&query=quiz or test
```

If you search on *quizortest*, for example, the server interprets it as a literal string and returns only exact matches.

### Request URL

```
http://server_name/api/xml
?action=sco-search
&query=querystring
&filter-definition=value
&sort-definition=value
&session=BreezeSessionCookieValue
```

### Parameters

Name	Type	Required	Description
query	Query string	Y	A string to search for. To use any of these special characters in the query string, escape them with a backslash before the character: + - &&     ! ( ) { } [ ] ^ " ~ * ? : \ The query string is not case-sensitive and allows wildcard characters * and ? at the end of the query string.
<a href="#">filter-definition</a>	Filter definition	N	A filter to reduce the volume of the response.
<a href="#">sort-definition</a>	Sort definition	N	A sort to return results in a certain sequence.
session	String	N	The value of the BREEZESESSION cookie. Use this parameter if you do not use a client-side cookie management library.

### Filters

You can filter the response on any element or attribute it contains.

### Response structure

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <sco-search-info>
    <sco sco-id=integer folder-id=integer type=allowedValue
      icon=allowedValue byte-count=integer tree-type=integer>
      <name>string</name>
      <url-path>string</url-path>
      <date-created>datetime</date-created>
      <date-modified>datetime</date-modified>
      <hit>integer</hit>
      <hit-type>allowedValue</hit-type>
      <thumbnail-path>string</thumbnail-path>
    </sco>
  </sco-search-info>
</results>
```



**Action reference**

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
sco-search-info		Container	The list of objects (SCOs) that match the search query.
sco		Container	Details about one object that matches the search.
	sco-id	Integer	The unique ID of the SCO.
	folder-id	Integer	The ID of the folder in which the SCO is stored.
	type	Allowed value	The content type assigned to the SCO (see <a href="#">type</a> for values).
	icon	Allowed value	The icon that visually identifies the SCO in a user interface.
	byte-count	Integer	The size of the SCO, in bytes.
	tree-type	Integer	The tree type.
name		String	The file name of the SCO.
url-path		String	The unique identifier that comes after the domain name in the SCO URL.
date-created		Datetime	The date the SCO was created.
date-modified		Datetime	The date the SCO was modified.
hit		Integer	The sequence number of this occurrence of the query string in the SCO.
hit-type		Allowed value	The type of content in which the search term was found. Allowed values are <code>metadata</code> and <code>slide</code> .
hit-url		String	A relative URL to the position where the search term was found in the content, for example, to a specific slide. Must be appended to the <code>url-path</code> .
thumbnail-path		String	A relative URL to an image of the SCO that contains the search term.

**Sample request**

`https://example.com/api/xml?action=sco-search&query=query`

**Action reference****Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <sco-search-info>
    <sco sco-id="5677964" folder-id="2562850" type="content"
      icon="producer" byte-count="5985" tree-type="13">
      <name>Final Quiz</name>
      <url-path>/p46125962/</url-path>
      <date-created>2005-05-09T14:24:36.390-07:00</date-created>
      <date-modified>2005-05-09T14:24:36.390-07:00</date-modified>
      <hit>0</hit>
      <hit-type>metadata</hit-type>
    </sco>
    <sco sco-id="5677964" folder-id="2562850" type="content"
      icon="producer" byte-count="5985">
      <name>Final Quiz</name>
      <url-path>/p46125962/</url-path>
      <date-created>2005-05-09T14:24:36.390-07:00</date-created>
      <date-modified>2005-05-09T14:24:36.390-07:00</date-modified>
      <hit>7</hit>
      <hit-type>slide</hit-type>
      <hit-url>slide=7</hit-url>
    </sco>
    ...
  </sco-search-info>
</results>
```

**sco-search-by-field****Availability**

Acrobat Connect Pro 7

**Description**

Provides a list of all SCOs matching the search text within the specified field. This action allows you to search for objects in the database based on the SCO's name, description, or author, or all three of those fields.

The `sco-search-by-field` action searches the content of some types of SCOs for the query string. The search includes folders, training courses, curriculums, meetings, content, and archives.

To search for multi-word terms with spaces between the words, search only on the first word in the term and use a wildcard at the end. For example, to search for *Sales Presentation*, use the following string:

```
query=sales*
```

**Note:** The `sco-search-by-field` command does not support the *and/or* operators.

**Request URL**

```
http://server_name/api/xml
?action=sco-search-by-field
&query=SearchTerm
&field=allowedValue
```

**Action reference**

**Parameters**

Name	Type	Required	Description
query	String	Y	The term to search for within the specified field. The query is case-insensitive.
field	String	N	The field to search. Accepts four possible values: <code>name</code> , <code>description</code> , <code>author</code> , or <code>allfields</code> : <ul style="list-style-type: none"> <li><code>name</code> searches only the name field for SCOs.</li> <li><code>description</code> searches only the description field for SCOs.</li> <li><code>author</code> matches the full name field (not the first-name or last-name fields) of the principal that created the SCOs.</li> <li><code>allfields</code> searches the name, description, and author fields.</li> </ul> If this parameter is omitted, the <code>name</code> field is searched.

**Filters**

Filters are supported on any field that can be returned. For example, you can use

```
&filter-gt-date-created=2007-09-12T08:00:00.000
```

if you want to show only results created after 8:00 AM on September 12, 2007.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <sco-search-info>
    <sco sco-id=integer tree-id=integer folder-id=integer type=allowedValue
status=allowedValue sco-data-id=integer source-sco-id=integer host-id=integer author-contact-
id=integer learning-time=allowedValue lang=allowedValue seq-id=integer icon=allowedValue
display-seq=integer max-retries=integer version=integer account-id=integer tree-type=integer>
    <name>string</name>
    <url-path>string</url-path>
    <date-created>datetime</date-created>
    <date-modified>datetime</date-modified>
    <principal-name>string</principal-name>
    <folder-name>string</folder-name>
  </sco>
</sco-search-info>
</results>
```

**Response value**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
sco-search-by-field-info		Container	The list of objects (SCOs) that match the search query.
sco		Container	Details about one object that matches the search.

**Action reference**

Element	Attribute	Type	Description
	sco-id	Integer	The unique ID of the SCO.
	folder-id	Integer	The ID of the folder in which the SCO is stored.
	type	Allowed value	The content type assigned to the SCO (see <a href="#">type</a> for values).
	icon	Allowed value	The icon that visually identifies the SCO in a user interface.
name		String	The file name of the SCO.
url-path		String	The unique identifier that comes after the domain name in the SCO URL.
date-created		Datetime	The date the SCO was created.
date-modified		Datetime	The date the SCO was last modified.
principal-name		String	The author of the SCO.
folder-name		String	The name of the folder in which the SCO is stored.

**Sample request**

`https://example.com/api/xml?action=sco-search-by-field&query=Marketing*&field=description`

**Sample response**

```
<results>
<status code="ok"/>
<sco-search-by-field-info>
  <sco sco-id="2007775205" tree-id="" folder-id="2007470298" type="meeting" status="" sco-
data-id="" source-sco-id="2007470292" host-id="" author-contact-id="" learning-time=""
lang="en" seq-id="" icon="virtual-classroom" display-seq="0" max-retries="" version="0"
account-id="2007470268" tree-type="4">
    <name>virt1</name>
    <url-path>/r72655596</url-path>
    <date-created>2007-10-10T16:41:31.643-07:00</date-created>
    <date-modified>2007-10-10T16:41:31.643-07:00</date-modified>
    <principal-name>Piet Pompies</principal-name>
    <folder-name>ppompies@adobe.com</folder-name>
  </sco>
  <sco sco-id="2007775257" tree-id="" folder-id="2007775254" type="folder" status="" sco-
data-id="" source-sco-id="" host-id="" author-contact-id="" learning-time="" lang="en" seq-
id="" icon="folder" display-seq="0" max-retries="" version="0" account-id="2007470268">
    <name>test1</name>
    <url-path>/f13818712</url-path>
    <date-created>2007-10-10T18:00:31.083-07:00</date-created>
    <date-modified>2007-10-10T18:00:31.083-07:00</date-modified>
    <principal-name>trainer two</principal-name>
    <folder-name>trainer@two.com</folder-name>
  </sco>
</sco-search-by-field-info>
</results>
```

## sco-shortcuts

### Availability

Breeze 4

### Description

Provides information about the folders relevant to the current user. These include a folder for the user's current meetings, a folder for the user's content, as well as folders above them in the navigation hierarchy.

To determine the URL of a SCO, concatenate the `url-path` returned by `sco-info`, `sco-contents`, or `sco-expanded-contents` with the `domain-name` returned by `sco-shortcuts`. For example, you can concatenate these two strings:

- `http://test.server.com` (the `domain-name` returned by `sco-shortcuts`)
- `/f2006123456/` (the `url-path` returned by `sco-info`, `sco-contents`, or `sco-expanded-contents`)

The result is this URL:

```
http://test.server.com/f2006123456/
```

You can also call `sco-contents` with the `sco-id` of a folder returned by `sco-shortcuts` to see the contents of the folder.

### Request URL

```
http://server_name/api/xml
    ?action=sco-shortcuts
    &session=BreezeSessionCookieValue
```

### Parameters

Name	Type	Required	Description
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

### Filters

Results cannot be filtered or sorted.

### Response structure

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=code />
  <shortcuts>
    <sco tree-id=integer sco-id=integer type=allowedValue>
      <domain-name>string</domain-name>
    </sco>
    ...
  </shortcuts>
</results>
```

**Action reference**

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
shortcuts		Container	Information about all of the folders that relate to the current user.
sco		Container	Information about one of the current user's folders.
	tree-id	Integer	The ID of the navigation tree that contains the folder. Several folders might have the same <code>tree-id</code> .
	sco-id	Integer	The unique ID of the folder.
	type	Allowed value	The type of the folder. Allowed values are shown in the following table.
domain-name		String	The domain name of the folder.

The values that can be returned in the `type` attribute of the `sco` element (for this call only, `sco-shortcuts`) identify Adobe Connect folders. Each folder type maps to a folder in Connect Central and requires certain permission levels to access, described in the following table.

Value of type	Description
account-custom	Customized content for an account, such as a customized login page, banner, and so on.
content	The Shared Content folder. Requires Administrator privilege or Manage permission.
courses	The Shared Training folder. Requires Administrator privilege or Manage permission.
events	The Shared Events folder. Requires Administrator privilege or Manage permission.
meetings	The Shared Meetings folder. Requires Administrator privilege or Manage permission.
my-courses	The My Training folder. By default, the individual user has Manage permission.
my-content	The My Content folder. By default, the individual user has Manage permission.
my-events	The My Events folder. By default, the individual user has Manage permission.
my-meetings	The My Meetings folder. By default, the individual user has Manage permission.
my-meeting-templates	The My Templates folder. By default, the individual user has Manage permission.
seminars	The Shared Seminars folder. Requires Administrator privilege or Manage permission.
shared-meeting-templates	The Shared Templates folder. Inherits permissions from Shared Meetings.
user-content	Contain the user content folders.
user-courses	Contain the user courses folders.
user-events	Contain the user events folders.
user-meetings	Contain the user meeting folders.

**Sample request**

`http://example.com/api/xml?action=sco-shortcuts`

**Action reference****Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <shortcuts>
    <sco tree-id="4930295" sco-id="2006258748" type="my-courses">
      <domain-name>http://example.com</domain-name>
    </sco>
    <sco tree-id="4930293" sco-id="2006258749" type="my-events">
      <domain-name>http://example.com</domain-name>
    </sco>
    ...
  </shortcuts>
</results>
```

**See also**

[sco-info](#), [sco-expanded-contents](#)

## sco-update

**Availability**

Breeze 4

**Description**

Creates metadata for a SCO, or updates existing metadata describing a SCO.

Call `sco-update` to create metadata only for SCOs that represent content, including meetings. You also need to upload content files with either [sco-upload](#) or Connect Central.

You must provide a `folder-id` or a `sco-id`, but not both. If you pass a `folder-id`, `sco-update` creates a new SCO and returns a `sco-id`. If the SCO already exists and you pass a `sco-id`, `sco-update` updates the metadata describing the SCO.

To create a course, pass `type=content&icon=course`, as in the following:

```
https://example.com/api/xml?action=sco-
update&name=AutomatedCourse&type=content&icon=course&folder-id=20002&source-sco-id=23510
```

After you create a new SCO with `sco-update`, call [permissions-update](#) to specify which users and groups can access it.

**Action reference**

**Request URL**

```

http://server_name/api/xml
  ?action=sco-update
  &author-info-1=string
  &author-info-2=string
  &author-info-3=string
  &date-begin=datetime
  &date-end=datetime
  &description=string
  &email=string
  &first-name=string
  &folder-id=integer
  &icon=allowedValue
  &lang=allowedValue
  &last-name=string
  &name=string
  &sco-id=integer
  &sco-tag=string
  &source-sco-id=integer
  &type=allowedValue
  &url-path=string
  &session=BreezeSessionCookieValue
    
```

**Parameters**

Name	Type	Required	Description
author-info-1	String	N	Information about the author. Used only with presentations. Can be used for the author's name or any other information.
author-info-2	String	N	Additional information about the author. Used only with presentations. Can be used for the author's professional title or any other information.
author-info-3	String	N	Additional information about the author. Used only with presentations. Can be used for the author's company name or any other information.
date-begin	Datetime	N	The scheduled beginning date and time, in <a href="#">ISO 8601</a> format. Used only for meetings and courses.
date-end	Datetime	N	The scheduled ending date and time, in <a href="#">ISO 8601</a> format. Used only for meetings and courses.
description	String	N	A description of the SCO to be displayed in the user interface.
email	String	N	The e-mail address of the contact person for a presentation (used only with presentation SCOs).
first-name	String	N	The first name of the contact person for a presentation (used only with presentation SCOs).
folder-id	Integer	Y/N	The ID of the folder in which a new SCO will be stored. Required for a new SCO, but do not use for an existing SCO.
lang	Allowed value	N	An abbreviation for the language associated with the SCO (see <a href="#">lang</a> for values). If not specified, the default value for the folder in which the SCO is created is used.
icon	Allowed value	N	The visual symbol used to identify a SCO in Connect Central; also provides information about the SCO in addition to its type.
last-name	String	N	The last name of the contact person for a presentation (used only with presentations).



**Action reference**

Name	Type	Required	Description
name	String	Y/N	The name of the SCO, with or without spaces. Required to create a SCO.
sco-id	Integer	Y/N	The unique ID of a SCO to update. Use <code>sco-id</code> or <code>folder-id</code> , but not both. Required to update an existing SCO.
sco-tag	String	N	A label for any information you want to record about a course. Use only with courses.
source-sco-id	Integer	N	The unique ID of a template you can use to create a meeting or a piece of content from which you can build a course.
type	Allowed value	N	The type of the new SCO (for allowed values, see <a href="#">type</a> ). The default value is <code>content</code> .
url-path	String	N	The custom part of the URL to the meeting room that comes after the domain name. The <code>url-path</code> must be unique within the folder. If not specified, the server assigns a value.
session	String	N	The value of the <code>BREEZESESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <sco folder-id=integer lang=allowedValue type=allowedValue
    sco-id=integer version=integer account-id=integer icon=integer>
    <url-path>string</url-path>
    <description>string</description>
    <name>string</name>
  </sco>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
sco		Container	Information about a new SCO just created, including the <code>sco-id</code> . Returned only if you create a SCO.
	folder-id	Integer	The ID of the folder in which the new SCO is stored.
	lang	Allowed value	A code for the language associated with the SCO (see <a href="#">lang</a> for values).
	type	Allowed value	The type of the new SCO (see <a href="#">type</a> for values).
	sco-id	Allowed value	The unique ID of the new SCO.
	version	Integer	The version number of the new SCO. When the SCO is first created, the <code>version</code> is 0.

**Action reference**

Element	Attribute	Type	Description
	account-id	Integer	The ID of the account in which the new SCO is created.
	icon	Integer	The type of icon that identifies a new SCO in Connect Central (see <a href="#">icon</a> for values).
url-path		String	The part of the SCO URL that comes after the domain name and uniquely identifies the SCO.
description		String	A text description of the SCO.
name		String	The name of the SCO.

**Sample request**

```
https://example.com/api/xml?action=sco-update&folder-id=2006258747
&description=test&name=More About Web Communities&type=content
&lang=en
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <sco folder-id="2006258747" account-id="624520" type="content" lang="en"
    icon="content" sco-id="2006752036" version="0">
    <url-path>p53884157</url-path>
    <description>test</description>
    <name>More About Web Communities</name>
  </sco>
</results>
```

**See also**

[sco-upload](#)

**sco-upload****Availability**

Breeze 4

**Description**

Uploads a file to the server and then builds the file, if necessary.

If you are adding a new file, call `sco-update` first and pass the `sco-id` returned to `sco-upload`. If you are updating the content of a file that already exists on the server, you can call `sco-upload` directly.

You must call `sco-upload` within an HTML `form` element. The `form` element must have an encoding type of `multipart/form-data`. The HTML form must also have an `input` element with `name=file`, as this example shows:

**Action reference**

```
<FORM action="http://domain-name/api/xml?action=sco-upload&sco-id=xx&summary=xxx&title=xxx"
  enctype="multipart/form-data" method="post">
  <P>
  What files are you sending?
  <INPUT type="file" name="file">
  <BR>
  <INPUT type="submit" value="Send"> <INPUT type="reset">
</FORM>
```

This form uploads a single file. To upload multiple files (for example, a PPT and a PPC file), you must use additional input elements with name=file, for example:

```
<FORM action="http://domain-name/api/xml?action=sco-upload&sco-
id=xxx&summary=xxx&title=xxxx" enctype="multipart/form-data" method="post">
  <P>
  PPT files you are sending <INPUT type="file" name="file"><BR>
  PPC files you are sending <INPUT type="file" name="file"><BR>
  <INPUT type="submit" value="Send"> <INPUT type="reset">
</FORM>
```

After the upload, call [sco-info](#) to get the status of the SCO. The status is initially in-progress, which means that the content is being built. When the status becomes active, the content build is finished, and users can access the content.

**Request URL**

```
http://server_name/api/xml
  ?action=sco-upload
  &file=formElementName
  &sco-id=integer
  &summary=string
  &title=string
  &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
file	Form element name	Y	The file to upload, sent from an input element with name=file in an HTML form. The HTML form must also have an encoding type of multipart/form-data defined in the form element.
sco-id	Integer	Y	The ID of the SCO you want to upload, returned by sco-update.
summary	String	N	A brief description of the SCO that Connect Central or your application displays.
title	String	N	The title of the SCO.
session	String	N	The value of the BREEZESSESSION cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Action reference****Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <files>
    <file>
      <path>string</path>
    </file>
  </files>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
files		Container	Information about all of the uploaded files. Deprecated and may be removed in a future release.
file		Container	Information about one file. Deprecated and may be removed in a future release.
path		String	The path to the newly uploaded file. For Adobe internal use only. Deprecated and may be removed in a future release.

**Sample request**

This request is created by uploading a file through an HTML form:

```
http://example.com/api/xml?action=sco-upload&sco-id=2006768386
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <files>
    <file>
      <path>624520/2006768386-10/input/WhatMakesAGreatFilm.ppt</path>
    </file>
  </files>
</results>
```

**See also**

[sco-update](#)

**user-accounts****Availability**

Breeze 4

**Action reference****Description**

Provides a list of the accounts a user belongs to.

The `user-accounts` action is only used when a user belongs to more than one account on the server and uses the same login ID and password for each. In that case, a user's login is likely to fail with a status message of `too-much-data`. This action is useful when you want to retrieve a list of a user's accounts and give the user a choice of which account to log in to.

**Request URL**

```
http://server_name/api/xml
  ?action=user-accounts
  &login=string
  &password=string
  &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
login	String	Y	The user's login name, which may be the user's e-mail address.
password	String	Y	The user's password.
session	String	N	The value of the <code>BREEZESESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted. The default sort is by `account-name`.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
  <users>
    <user user-id=integer account-id=integer>
      <name>string</name>
      <date-expired>datetime</date-expired>
    </user>
    ...
  </users>
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
<a href="#">status</a>		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).
users		Container	Information about the accounts the user belongs to.
user		Container	Information about a user and an account.
	user-id	Integer	The ID of the user on the server.

**Action reference**

Element	Attribute	Type	Description
	account-id	Integer	The ID of the account the user belongs to.
name		String	The name of the account the user belongs to.
date-expired		Datetime	The date and time the user's login expires.

**Sample request**

```
https://sample.com/api/xml?action=user-accounts&login=joy@acme.com
&password=bigdog
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <users>
    <user user-id="2006258745" account-id="624520">
      <name>Test Account</name>
      <date-expired>2999-12-31T16:00:00.000-08:00</date-expired>
    </user>
  </users>
</results>
```

**user-transcript-update****Availability**

Breeze 5

**Description**

Overrides the score on an item within a curriculum.

For example, you can use `user-transcript-update` to give a user a score for an external training. This action works only for items within a curriculum, and you need `manage` permission for the curriculum.

**Request URL**

```
http://server_name/api/xml
?action=user-transcript-update
&curriculum-id=integer
&sco-id=integer
&status=allowedValue
&score=integer
&principal-id=integer
&session=BreezeSessionCookieValue
```

**Action reference**

**Parameters**

Name	Type	Required	Description
curriculum-id	Integer	N	The ID of the curriculum.
sco-id	Integer	Y	The unique ID of a SCO with a score you want to override.
status	Allowed value	Y	A value showing the status of the user's attempt to use this SCO. Allowed values are <code>completed</code> , <code>incomplete</code> , <code>user-passed</code> , <code>user-failed</code> , and <code>not-attempted</code> .
score	Integer	Y	An integer value that represents the score the user has attained on this SCO.
principal-id	Integer	Y	The ID of the user whose transcript will be overridden.
session	String	N	The value of the <code>BREEZESESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
</results>
```

**Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Sample request**

```
https://example.com/api/xml?action=user-transcript-update
&curriculum-id=2006298444&sco-id=2006298445&status=user-passed
&principal-id=2006258745&score=100
```

**Sample response**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
</results>
```

**See also**

[learning-path-info](#), [learning-path-update](#)

**Action reference****user-update-pwd****Availability**

Breeze 4

**Description**

Changes a user's password. A password can be changed in either of these cases:

- By an Administrator logged in to the account, with or without the user's old password
- By any Adobe Connect Server user, with the user's `principal-id` number, login name, and old password

An Administrator can create rules for valid passwords on the server. These rules might include, for example, the number and types of characters a password must contain. If a user submits a new password that does not adhere to the rules, Adobe Connect would throw an error showing that the new password is invalid.

When you call `user-update-pwd`, the password is sent over HTTP or HTTPS in hashed form.

**Request URL**

```
http://server_name/api/xml
  ?action=user-update-pwd
  &user-id=integer
  &password-old=string
  &password=string
  &password-verify=string
  &session=BreezeSessionCookieValue
```

**Parameters**

Name	Type	Required	Description
<code>user-id</code>	Integer	Y	The ID of the user.
<code>password-old</code>	String	Y/N	The user's current password. Required for regular users, but not for Administrator users.
<code>password</code>	String	Y	The new password.
<code>password-verify</code>	String	Y	A second copy of the new password, for verification.
<code>session</code>	String	N	The value of the <code>BREEZESSESSION</code> cookie. Use this parameter if you do not use a client-side cookie management library.

**Filters**

Results cannot be filtered or sorted.

**Response structure**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code=allowedValue />
</results>
```



**Action reference****Response values**

Element	Attribute	Type	Description
results		Container	All results the action returns.
status		Empty, with attributes	The status of the response.
	code	Allowed value	A code indicating the response status (see <a href="#">status</a> ).

**Sample request**

This request can be used by an Administrator to change a user's password without knowing the old password:

```
https://example.com/api/xml?action=user-update-pwd&user-id=12345&password=newone&password-verify=newone
```

**Sample response**

This response shows that the change was successful:

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
</results>
```

# Chapter 8: Filter and sort reference

This chapter is a reference for filters and sort values you use to reduce the volume of the response from XML actions in Adobe® Connect™ Web Services.

## filter-definition

### Description

A filter is a special type of parameter that reduces the volume of the response. When you see *filter-definition* in a request URL syntax, substitute a filter definition.

To create a filter definition, start with the keyword `filter`, add an modifier (if desired), then a field name (if allowed), and then a value, using this syntax:

```
filter-modifier-field=value
```

The modifiers you can add are listed in the following table.

Filter	Description
<code>filter-field=value</code>	Returns all items for which the data in <i>field</i> exactly matches <i>value</i> .
<code>filter-like-field=value</code>	Returns all items with the string <i>value</i> within <i>field</i> , even if <i>field</i> is not an exact match.
<code>filter-out-field=value</code>	Filters out or excludes any items with <i>value</i> in <i>field</i> .
<code>filter-rows=value</code>	Limits the results to the number of rows specified in <i>value</i> .
<code>filter-start=value</code>	Starts the results at the index number specified in <i>value</i> .
<code>filter-gt-datefield=value</code>	Selects all items with a date after <i>value</i> . Works only with date fields. The value must be a date in <a href="#">ISO 8601</a> format.
<code>filter-lt-datefield=value</code>	Selects all items with a date earlier than <i>value</i> . Works only with date fields. The value must be a date in <a href="#">ISO 8601</a> format.
<code>filter-gte-datefield=value</code>	Selects all items with a value in <i>field</i> greater than or equal to <i>value</i> . Works only with date fields. The value must be a <i>date</i> in <a href="#">ISO 8601</a> format.
<code>filter-lte-datefield=value</code>	Selects all items with a value in <i>field</i> less than or equal to <i>value</i> . Works only with dates. The date uses <a href="#">ISO 8601</a> format.
<code>filter-is-member=value</code>	Selects all principals that are members of a group, specified in a separate parameter. Takes a Boolean value of <code>true</code> or <code>false</code> .

The value is case insensitive. For example, either of these filters matches a meeting with the name *August All Hands Meeting*:

```
&filter-name=August All Hands Meeting
&filter-name=august all hands meeting
```

Some modifiers require a field name on which to filter results, for example, `name`. Other filters do not take a field name. For those filters that accept field names, the allowed fields vary for different actions. Check a specific action in “[Action reference](#)” on page 58 to learn which field names you can use in filters.

**Filter and sort reference****Exact match filter**

```
filter-name=Goals Review
```

Matches items with *Goals Review* (or any mixed case pattern of the same string) in the name.

**Similar match filter**

```
filter-like-name=Goals
```

Matches any item that includes *Goals* (or any mixed case pattern of the same string) in the name, including *Goals Review* and *Quarterly Goals*.

**Exclude items filter**

```
filter-out-name=Status
```

Excludes all items with *Status* (or any mixed case pattern of the same string) in the name.

**Match and exclude items**

```
filter-like-name=Goals&filter-out-status=active
```

Matches any item with *Goals* (or any mixed case pattern of the same string) in the name that is no longer active.

**Match a start date**

```
filter-gt-date-begin=2005-05-01&sort-name=asc
```

Matches any item with a start date of May 1, 2005, sorting the items in ascending order by name.

**Match a date range**

```
filter-gt-date-begin=2005-05-01&filter-lt-date-begin=2005-05-31
```

Returns all items with a start date after May 1, 2005 and before May 31, 2005.

**See also**

[sort-definition](#)

## sort-definition

When you see *sort-definition* in a request URL in this reference, create a sort filter with a field name and a value describing how you want the results sorted, in this syntax:

```
sort-field=value
```

Replace *sort* with any of these exact values: `sort` (for a single sort), `sort1` (for the primary sort of two), or `sort2` (for a secondary sort on the results returned by `sort1`).

The *field* variable defines the field you are sorting on. The fields you can use vary by call, so check the API reference for the call you are making.

The *value* is always `asc` (for *ascending*) or `desc` (for *descending*), defining the sequence of the results. Putting this all together, the parts of a sort filter are shown in the following table:

**Filter and sort reference**

Sort Type	Field	Value	Description
sort	Vary by call.	asc or desc	Sort results by the specified field, in either ascending or descending order.
sort1	Vary by call.	asc or desc	Sort results by a field, either ascending or descending, and then pass the results to the next sort.
sort2	Vary by call.	asc or desc	When results returned by the primary sort are equal, such as same name or group, do a secondary sort by the specified field in either ascending or descending order.

Simple examples of *sort-field=value*, with one level of sort, look like this:

```
sort-name=asc
sort-date=desc
```

The following table gives you more detail on how the sort values *asc* and *desc* work:

Value	Description
asc	Ascending order. For alphabetical lists, begin with A and end with Z. For lists ordered by number or date, start with lowest number or earliest date.
desc	Descending order. For alphabetical lists, begin with Z and end with A. For lists ordered by number or date, start with highest number or most recent date.

Your results may call for using both primary and secondary sorts with *sort1* and *sort2*. For example, when calling *principal-list* to list principals, you can do a primary sort on the *type* field, and then a secondary sort on the *name* field (this way, all principals of a specific type are grouped together and then sorted by name in each group).

You would specify two levels of sort like this:

```
sort1-type=asc&sort2-name=desc
```

**See also**

[filter-definition](#)

# Chapter 9: Using the Telephony XML API

This chapter is a reference for the telephony XML API you use to call back to Adobe® Connect™ Web Services. This API is available in Adobe Connect 7.5 Service Pack 1 and later.

The response values from the XML API actions represent name/value pairs that are fields in the Adobe Connect account. A Java Row object is a name/value pair instantiated from the XML response string. For more information, see *Adobe Connect Telephony API Javadoc* at [www.adobe.com/go/learn\\_cnn\\_telephony\\_javadoc\\_en](http://www.adobe.com/go/learn_cnn_telephony_javadoc_en).

If an API can't run as designed (for example, a mandatory parameter isn't passed), `<status code="no-data"/>` is returned.

The following table summarizes the XML API actions available, in alphabetical order:

Name	Description
<a href="#">telephony-profile-delete</a>	Deletes the specified profile and removes it from any meetings with which it is associated.
<a href="#">telephony-profile-info</a>	Retrieves the settings for a specified profile.
<a href="#">telephony-profile-list</a>	Retrieves the profiles associated with the specified user's principal ID.
<a href="#">telephony-profile-update</a>	Creates a new telephony profile or updates an existing profile.
<a href="#">telephony-provider-conf-number-update</a>	For the specified provider, creates conference numbers that can be used for dialing in to an audio conferences.
<a href="#">telephony-provider-delete</a>	Deletes a telephony provider and all information associated with that provider.
<a href="#">telephony-provider-dial-in-info-update</a>	Creates or updates the dial-in-sequence for a provider.
<a href="#">telephony-provider-field-delete</a>	Deletes a provider field.
<a href="#">telephony-provider-field-list</a>	Displays a list of the fields of a provider.
<a href="#">telephony-provider-field-update</a>	Creates or updates a provider field.
<a href="#">telephony-provider-info</a>	Displays information on a telephony provider.
<a href="#">telephony-provider-list</a>	Returns a list of telephony providers.
<a href="#">telephony-provider-update</a>	Creates a new provider or updates an existing provider.

## telephony-profile-delete

Deletes the specified profile and removes it from any meetings with which it is associated. To determine profile names, see “[telephony-profile-list](#)” on page 219.

### Syntax

```
http://<connect_server_name>/api/xml?action=telephony-profile-delete&profile-id=integer
```

### Parameters

Name	Required?	Description
profile-id	Y	Integer that specifies the ID of the profile you want to delete.

### Permission

You must have permission to modify the profile you want to delete.

### Example

Request:

```
http://connectdev1/api/xml?action=telephony-profile-delete&profile-id=11066
```

Response:

```
<results>  
  <status code="ok"/>  
>  
</results>
```

## telephony-profile-info

Retrieves the settings for a specified profile.

### Syntax

```
http://server_name/api/xml?action=telephony-profile-info&profile-id=integer
```

### Parameters

Name	Required?	Description
account-id	N	Integer that specifies the ID of the account to which the profile belongs. If not specified, the logged in user's account is used.
profile-id	Y	Integer that specifies the ID of the profile you want to view.

### Permission

You must have permission to view the profile.

### Example

Request:

```
http://connectdev1.corp.adobe.com/api/xml?action=telephony-profile-info&profile-id=11422
```

Response:

```
<results>
  <status code="ok"/>
  -
  <telephony-profile profile-id="11422" provider-id="11319" profile-status="enabled"
provider-type="integrated">
  <adaptor-id>premiere-adaptor</adaptor-id>
  <provider-name>premiere-adaptor</provider-name>
  -
  <class-name>
  com.macromedia.breeze_ext.premiere.gateway.PTekGateway
  </class-name>
  <profile-name>PNA1-1</profile-name>
  <provider-status>enabled</provider-status>
</telephony-profile>
-
  <telephony-profile-fields disabled="" principal-id="11202" profile-id="11422" profile-
status="enabled" provider-id="11319">
  <profile-name>PNA1-1</profile-name>
  <x-tel-premiere-conference-id>5074202</x-tel-premiere-conference-id>
  <x-tel-premiere-conference-number>1-888-208-8183</x-tel-premiere-conference-number>
  <x-tel-premiere-uv-conference-number>1-888-208-8183</x-tel-premiere-uv-conference-
number>
  <x-tel-premiere-participant-code>726988</x-tel-premiere-participant-code>
  <x-tel-premiere-user-id>7003155</x-tel-premiere-user-id>
  <x-tel-premiere-password>#C$F%P@E!i4/XRiuxhMAZLnQPpS4f0w==</x-tel-premiere-password>
  <x-tel-premiere-moderator-code>7269889</x-tel-premiere-moderator-code>
  </telephony-profile-fields>
</results>
```

## telephony-profile-list

Retrieves the profiles associated with the specified user's principal ID.

### Syntax

`http://server_name/api/xml?action=telephony-profile-list&principal-id=integer`

### Parameters

Name	Required?	Description
principal-id	N	Integer that specifies the principal ID of the user for whom profiles are retrieved. If you don't specify a value, the logged-in user's principal ID is used.

### Permission

You must have view permissions on the principal-id, if provided.

### Example

Request:

`http://connectdev1/api/xml?action=telephony-profile-list&principal-id=11032`

Response:

```
<results>
  <status code="ok"/>
  -
  <telephony-profiles>
  -
    <profile profile-id="11091" provider-id="11049" profile-status="enabled">
      <adaptor-id>premiere-emea-adaptor</adaptor-id>
      <name>Premiere EMEA</name>
      <profile-name>PE1</profile-name>
    </profile>
  -
    <profile profile-id="11232" provider-id="11035" profile-status="enabled">
      <adaptor-id>premiere-adaptor</adaptor-id>
      <name>Premiere NA</name>
      <profile-name>P2</profile-name>
    </profile>
  </telephony-profiles>
</results>
```

## telephony-profile-update

Creates a new telephony profile or updates an existing profile.

### Syntax

`http://server_name/api/xml?action=telephony-profile-update&principal-id=integer&profile-name=profile-name&profile-status=profile-status&field-id=field-id&value=value&provider-id=integer`

### Parameters

Name	Required?	Description
conf-number	N	String that specifies the conference number associated with this profile. If you provide a value, any existing conference numbers for this profile are deleted.
field-id	N	String that specifies the field whose value needs to be updated. If this value is specified, you must also specify a value for provider-id.
location	Y if you specify a value for conf-number; otherwise N	String specifying the country code (for example, UK) of the location to be updated.
principal-id	N	Integer that specifies the user for which the profile is created or updated. If not specified, the principal ID of the user who is currently logged in is used.
profile-id	Y if you are updating a profile	Integer that specifies the profile to be updated. If not specified, a new profile is created.
profile-name	Y if you are creating a profile	The name of the profile being created or updated.



Name	Required?	Description
profile-status	Y if you are updating a profile	String that specifies the status of the profile. Acceptable values are <code>enabled</code> and <code>disabled</code> . If you disable a profile, all of its associations with meetings are removed. If you are creating a new profile, the default value is <code>enabled</code> .
provider-id	Y if you are creating a profile	String that specifies the telephony provider for the profile being created or updated
value	Y if field-id is specified; otherwise N	Specifies the value of the field-id.

### Permission

If you are creating a new profile, you must have modify permissions on the principal-id. If you are updating a profile, you must have modify permissions on the profile.

### Example

Request (to update a profile):

```
http://connectdev1/api/xml?action=telephony-profile-update&principal-id=11032&profile-id=11091&profile-status=disabled
```

Response:

```
<results>
  <status code="ok"/>
</results>
```

Request (to create a profile):

```
http://connectdev1/api/xml?action=telephony-profile-update&principal-id=11032&profile-name=P3&profile-status=enabled&field-id=x-tel-premiere-emea-user-id&value=8073174&field-id=x-tel-premiere-emea-moderator-code&value=4963832&field-id=x-tel-premiere-emea-password&value=password&provider-id=11049
```

Response:

```
<results>
  <status code="ok"/>
  -
  <profile profile-status="enabled" provider-id="11050" principal-id="11032" profile-id="11900">
    <profile-name>PE3</profile-name>
  </profile>
</results>
```

## telephony-provider-conf-number-update

For the specified provider, creates conference numbers that can be used for dialing in to an audio conferences.

*Note: This API deletes any existing conference numbers.*

### Syntax

```
http://server_name/api/xml?action=telephony-provider-conf-number-update&provider-id=integer&conf-number=integer&location=location
```

### Parameters

Name	Required?	Description
provider-id	Y	Integer specifying the provider ID for which the conference number is to be updated.
conf-number	Y	Integer specifying the conference number to be updated.
location	Y	String specifying the country code (for example, UK) of the location to be updated.

### Permission

You must have permission to modify the provider whose conference number is being updated.

### Example

Request:

```
http://connectdev1/api/xml?action=telephony-provider-conf-number-update&provider-id=11712&conf-number=4567&location=USA
```

Response:

```
<results>  
  <status code="ok"/>  
</results>
```

## telephony-provider-delete

Deletes a telephony provider and all information associated with that provider.

### Syntax

```
http://server_name/api/xml?action=telephony-provider-delete&provider-id=integer
```

### Parameters

Name	Required?	Description
provider-id	Y	Integer specifying the provider ID for the provider you want to delete.

### Permission

You must have permission to modify the provider which is being deleted.

### Example

Request:

```
http://connectdev1/api/xml?action=telephony-provider-delete&provider-id=11049
```

Response:

```
<results>  
  <status code="ok"/>  
</results>
```

## telephony-provider-dial-in-info-update

Creates or updates the dial-in-sequence for a provider.

*Note: This API deletes any existing dial-in sequences.*

### Syntax

```
http://server_name/api/xml?action=telephony-provider-dial-in-info-udpate&step-type=step-type&name=name&provider-id=integer&field-id=integer&value=value
```

### Parameters

Name	Required?	Description
step-type	Y	String that specifies the type of step that is being updated or added. Acceptable values are <code>conf-num</code> , <code>dtmf</code> , and <code>delay</code> .
field-id	N	Integer that specifies the field whose value needs to be updated.
value	N	Specifies the value of the specified field-id.
name	N	String that specifies the name of the dial-in step being created.
provider-id	Y	Integer that specifies the ID of the provider whose dial-in step being created.

### Permission

You must have modify permission on the provider whose dial-in-sequence is being update or created.

### Example

Request:

```
http://connectdev1/api/xml?action=telephony-provider-dial-in-info-udpate&step-type=conf-num&name=xyz&provider-id=11814&field-id=11058&value=1000
```

Response:

```
<results>  
  <status code="ok"/>  
</results>
```

## telephony-provider-field-delete

Deletes a provider field. For a list of available fields, see “[telephony-provider-field-list](#)” on page 224.

### Syntax

```
http://server_name/api/xml?action=telephony-provider-field-delete&provider-id=integer&xml-name=xml-name&field-id=integer
```

### Parameters

Name	Required?	Description
provider-id	Y	Integer that specifies the provider whose field is to be deleted.
field-id	N if xml-name is specified, otherwise Y	Integer that specifies the ID of the field to be deleted.
xml-name	N if field-id is specified, otherwise Y	String that specifies the XML name of the field to be deleted. If you also specify a value for field-id, this value is ignored.

### Permission

You must have modify permission on the provider whose field is being deleted

### Example

Request:

```
http://connectdev1/api/xml?action=telephony-provider-field-delete&provider-id=11814&xml-name=x-tel-premiere-emea-uv-conference-number
```

Response:

```
<results>  
  <status code="ok"/>  
</results>
```

## telephony-provider-field-list

Displays a list of the fields of a provider.

### Syntax

```
http://server_name/api/xml?action=telephony-provider-field-list&provider-id=integer
```

### Parameters

Name	Required?	Description
provider-id	Y	The provider whose fields are to be listed.

### Permission

You must have view permissions on the provider.

### Example

Request:

```
http://connectdev1/api/xml?action=telephony-provider-field-list&provider-id=11718
```

Response:

```
<results>
  <status code="ok"/>
  -
  <telephony-provider-fields>
  -
    <field provider-id="11718" field="11038" field-id="x-premiere-direct-phone" display-
in-meeting="none" required="false" user-specified="true" input-type="text" is-hidden="true">
      <name>Phone Number</name>
    </field>
  -
    <field provider-id="11718" field="11039" field-id="x-premiere-direct-phone-key"
display-in-meeting="none" required="false" user-specified="false" input-type="text" is-
hidden="true">
      <name>{x-premiere-direct-phone-key}</name>
    </field>
  -
    <field provider-id="11718" field="11046" field-id="x-tel-premiere-sign-up-text"
display-in-meeting="none" required="false" user-specified="false" input-type="text" is-
hidden="false">
      -
      <name>
        To learn more about Premiere Global or to sign up for a new account, please go
to
        <u><a target="_blank"
href="http://www.premierglobal.com/adobeconnect/">http://www.premierglobal.com/adobeconnec
t</a></u>.
      </name>
    </field>
  -
    <field provider-id="11718" field="11051" field-id="x-tel-premiere-emea-conference-
number-part2" display-in-meeting="participants" required="false" user-specified="false"
input-type="textarea" is-hidden="false">
      <name>{x-tel-premiere-emea-conference-number-part2}</name>
    </field>
  -
    <field provider-id="11718" field="11052" field-id="x-tel-premiere-emea-conference-
number-part3" display-in-meeting="participants" required="false" user-specified="false"
input-type="textarea" is-hidden="false">
      <name>{x-tel-premiere-emea-conference-number-part3}</name>
    </field>
  -
    <field provider-id="11718" field="11053" field-id="x-tel-premiere-emea-conference-
number-part4" display-in-meeting="participants" required="false" user-specified="false"
input-type="textarea" is-hidden="false">
      <name>{x-tel-premiere-emea-conference-number-part4}</name>
    </field>
  -
    <field provider-id="11718" field="11054" field-id="x-tel-premiere-emea-conference-
number-part5" display-in-meeting="participants" required="false" user-specified="false"
input-type="textarea" is-hidden="false">
      <name>{x-tel-premiere-emea-conference-number-part5}</name>
    </field>
  -
    <field provider-id="11718" field="11055" field-id="x-tel-premiere-emea-conference-
number-part6" display-in-meeting="participants" required="false" user-specified="false"
input-type="textarea" is-hidden="false">
      <name>{x-tel-premiere-emea-conference-number-part6}</name>
```

```
</field>
-
  <field provider-id="11718" field="11056" field-id="x-tel-premiere-emea-conference-id"
display-in-meeting="none" required="false" user-specified="false" input-type="text" is-
hidden="true">
  <name>{x-tel-premiere-emea-conference-id}</name>
</field>
-
  <field provider-id="11718" field="11057" field-id="x-tel-premiere-emea-conference-
number" display-in-meeting="participants" required="false" user-specified="false" input-
type="textarea" is-hidden="false">
  <name>Conference Number(s)</name>
</field>
-
  <field provider-id="11718" field="11058" field-id="x-tel-premiere-emea-uv-conference-
number" display-in-meeting="none" required="false" user-specified="false" input-type="text"
is-hidden="true">
  <name>{x-tel-premiere-emea-uv-conference-number}</name>
</field>
-
  <field provider-id="11718" field="11059" field-id="x-tel-premiere-emea-participant-
code" display-in-meeting="participants" required="false" user-specified="false" input-
type="text" is-hidden="false">
  <name>Participant Code</name>
</field>
-
  <field provider-id="11718" field="11060" field-id="x-tel-premiere-emea-user-id"
display-in-meeting="none" required="true" user-specified="true" input-type="text" is-
hidden="false">
  <name>Client ID</name>
</field>
-
  <field provider-id="11718" field="11061" field-id="x-tel-premiere-emea-password"
display-in-meeting="none" required="true" user-specified="true" input-type="password" is-
hidden="false">
  <name>Premiere Password</name>
</field>
-
  <field provider-id="11718" field="11062" field-id="x-tel-premiere-emea-moderator-
code" display-in-meeting="hosts" required="true" user-specified="true" input-type="text" is-
hidden="false">
  <name>Moderator Code</name>
</field>
</telephony-provider-fields>
</results>
```

## telephony-provider-field-update

Creates or updates a provider field. If updating a field might result in profiles for this provider to be invalid, all profiles are disassociated from meetings.

For example, if you change a field from optional to mandatory, some existing profiles might not meet the new criteria. Therefore, Connect disassociates all this provider's profiles from meetings. After updating the field or fields, you need to update profiles as needed and then re-associate them with meetings.

### Syntax

`http://server_name/api/xml?action=telephony-provider-field-update&provider-id=integer&input-type=input-type&display-in-meeting=display-in-meeting&required=boolean&user-specified=boolean&is-hidden=boolean`

### Parameters

Name	Required?	Description
provider-id	Y	Integer that specifies the provider whose fields need to be updated or created.
field-id	Y if you don't specify a value for xml-name; otherwise N	Integer that specifies the ID of the field to be updated or created.
xml-name	Y if you don't specify a value for field-id; otherwise N	String that specifies the XML name of the field to be updated or created. If a value is also specified for field-id, xml-name is ignored.
input-type	Y	Input type of this field. Acceptable values are <code>text</code> , <code>password</code> , <code>textarea</code> , and <code>url</code> .
display-in-meeting	N	String that specifies which participants can see this field in the Connect meeting room. Acceptable values are <code>none</code> and <code>participants</code> .
required	Y	Boolean value that specifies if this is a required field.
user-specified	N	Boolean value that specifies whether this is a field that the user specifies, such as password.
is-hidden	N	Boolean value that specifies whether this field can be displayed through a user interface.

### Permission

You must have permission to modify the provider whose field you are creating or updating.

### Example

Request (to create a provider field):

```
http://connectdev1/api/xml?action=telephony-provider-field-update&provider-id=11718&input-type=text&display-in-meeting=participants&required=true&user-specified=true&is-hidden=false
```

Response:

```
<results>
  <status code="ok"/>
  -
  <telephony-provider provider-status="enabled" provider-type="user-conf" provider-id="12000">
    <name>my-provider</name>
    <adaptor-id>12000-adaptor</adaptor-id>
  </telephony-provider>
</results>
```

Request (to update a provider field):

```
http://connectdev1/api/xml?action=telephony-provider-field-update&provider-id=11718&input-type=text&display-in-meeting=participants&required=false&user-specified=false&is-hidden=false&field-id=11829&xml-name=x-tel-11829
```

Response:

```
<results>  
  <status code="ok"/>  
</results>
```

## telephony-provider-info

Displays information on a telephony provider, including dial-in sequence, provider fields, and associated conference numbers.

### Syntax

```
http://server_name/api/xml?action=telephony-provider-info&is-meeting-host=boolean&provider-id=integer&account-id=integer&principal-id=integer
```

### Parameters

Name	Required?	Description
provider-id	Y	Integer that specifies the provider ID for which information is to be retrieved.
principal-id	N	Integer which, if specified, represents the principal ID of the user who created the provider. If not specified, this value is the principal ID of the currently logged-in user.
account-id	N	If provider-id represents an account-level provider (that is, a provider that was not created by a user), this value is a string that specifies the ID of the account to which the provider belongs. If not specified, this value is the ID of the account of the logged-in user.
is-meeting-host	N	A Boolean value that specifies whether provider-id is a user-configured provider. If this value is <code>true</code> , information is returned based on the value provided for principal-id.

### Permission

For a user-configured provider, you must have view permissions on the provider. For account-level providers, you must have view permission on the account.

### Example

Request:

```
http://connectdev1/api/xml?action=telephony-provider-info&is-meeting-host=false&provider-id=11718&account-id=7
```

Response:



```
<results>
  <status code="ok"/>
  -
  <telephony-provider provider-id="11718" provider-type="integrated" token-length="0">
    -
    <class-name>
      com.macromedia.breeze_ext.premiere.gateway.EMEA.PTekGateway
    </class-name>
    <adaptor-id>premiere-emea-adaptor</adaptor-id>
    <name>Premiere EMEA</name>
    <token-prefix>#4</token-prefix>
    <token-postfix>#</token-postfix>
    <provider-status>enabled</provider-status>
  </telephony-provider>
  -
  <telephony-provider-dial-in-info>
    <step provider-id="11718" sequence-number="1" field="11058" step-type="conf-num"/>
    -
    <step provider-id="11718" sequence-number="2" field="" step-type="delay">
      <value>6000</value>
    </step>
    <step provider-id="11718" sequence-number="3" field="11059" step-type="dtmf"/>
    -
    <step provider-id="11718" sequence-number="4" field="" step-type="dtmf">
      <value>#</value>
    </step>
    -
    <step provider-id="11718" sequence-number="5" field="" step-type="delay">
      <value>5000</value>
    </step>
  </telephony-provider-dial-in-info>
  -
  <telephony-provider-fields>
    -
    <field provider-id="11718" field="11038" field-id="x-premiere-direct-phone" display-
in-meeting="none" required="false" user-specified="true" input-type="text" is-hidden="true">
      <name>Phone Number</name>
    </field>
    -
    <field provider-id="11718" field="11039" field-id="x-premiere-direct-phone-key"
display-in-meeting="none" required="false" user-specified="false" input-type="text" is-
hidden="true">
      <name>{x-premiere-direct-phone-key}</name>
    </field>
    -
    <field provider-id="11718" field="11046" field-id="x-tel-premiere-sign-up-text"
display-in-meeting="none" required="false" user-specified="false" input-type="text" is-
hidden="false">
      -
      <name>
        To learn more about Premiere Global or to sign up for a new account, please go
to
        <u><a target="_blank"
href="http://www.premierglobal.com/adobeconnect/">http://www.premierglobal.com/adobeconnec
t</a></u>.
      </name>
    </field>
```

```
-
  <field provider-id="11718" field="11051" field-id="x-tel-premiere-emea-conference-
number-part2" display-in-meeting="participants" required="false" user-specified="false"
input-type="textarea" is-hidden="false">
    <name>{x-tel-premiere-emea-conference-number-part2}</name>
  </field>
-
  <field provider-id="11718" field="11052" field-id="x-tel-premiere-emea-conference-
number-part3" display-in-meeting="participants" required="false" user-specified="false"
input-type="textarea" is-hidden="false">
    <name>{x-tel-premiere-emea-conference-number-part3}</name>
  </field>
-
  <field provider-id="11718" field="11053" field-id="x-tel-premiere-emea-conference-
number-part4" display-in-meeting="participants" required="false" user-specified="false"
input-type="textarea" is-hidden="false">
    <name>{x-tel-premiere-emea-conference-number-part4}</name>
  </field>
-
  <field provider-id="11718" field="11054" field-id="x-tel-premiere-emea-conference-
number-part5" display-in-meeting="participants" required="false" user-specified="false"
input-type="textarea" is-hidden="false">
    <name>{x-tel-premiere-emea-conference-number-part5}</name>
  </field>
-
  <field provider-id="11718" field="11055" field-id="x-tel-premiere-emea-conference-
number-part6" display-in-meeting="participants" required="false" user-specified="false"
input-type="textarea" is-hidden="false">
    <name>{x-tel-premiere-emea-conference-number-part6}</name>
  </field>
-
  <field provider-id="11718" field="11056" field-id="x-tel-premiere-emea-conference-id"
display-in-meeting="none" required="false" user-specified="false" input-type="text" is-
hidden="true">
    <name>{x-tel-premiere-emea-conference-id}</name>
  </field>
-
  <field provider-id="11718" field="11057" field-id="x-tel-premiere-emea-conference-
number" display-in-meeting="participants" required="false" user-specified="false" input-
type="textarea" is-hidden="false">
    <name>Conference Number(s)</name>
  </field>
-
  <field provider-id="11718" field="11058" field-id="x-tel-premiere-emea-uv-conference-
number" display-in-meeting="none" required="false" user-specified="false" input-type="text"
is-hidden="true">
    <name>{x-tel-premiere-emea-uv-conference-number}</name>
  </field>
-
  <field provider-id="11718" field="11059" field-id="x-tel-premiere-emea-participant-
code" display-in-meeting="participants" required="false" user-specified="false" input-
type="text" is-hidden="false">
    <name>Participant Code</name>
  </field>
```

```
-
  <field provider-id="11718" field="11060" field-id="x-tel-premiere-emea-user-id"
display-in-meeting="none" required="true" user-specified="true" input-type="text" is-
hidden="false">
  <name>Client ID</name>
</field>
-
  <field provider-id="11718" field="11061" field-id="x-tel-premiere-emea-password"
display-in-meeting="none" required="true" user-specified="true" input-type="password" is-
hidden="false">
  <name>Premiere Password</name>
</field>
-
  <field provider-id="11718" field="11062" field-id="x-tel-premiere-emea-moderator-
code" display-in-meeting="hosts" required="true" user-specified="true" input-type="text" is-
hidden="false">
  <name>Moderator Code</name>
</field>
</telephony-provider-fields>
</results>
```

## telephony-provider-list

Returns a list of telephony providers.

- To return a list of user-configured providers for a particular user, pass the user's principal ID to `principal-id` and pass a value of `true` for `is-meeting-host`.
- To return a list of account-level providers for a specified account, pass the account's ID to `account-id` and pass a value of `false` for `is-meeting-host`.
- To return a list of account-level providers for the account of the currently logged-in user, pass a value of `false` for `is-meeting-host`.

### Syntax

`http://server_name/api/xml?action=telephony-provider-list&is-meeting-host=boolean&account-id=integer&principal-id=integer`

### Parameters

Name	Required?	Description
principal-id	Y if the value of <code>is-meeting-host</code> is <code>true</code> ; otherwise N	Integer that represents the principal ID of the user who created the user-configured providers. If not specified, the principal ID of the logged-in user is used.
account-id	N	Integer that specifies the ID of the account to which the account-level providers belong. If not specified, the account ID of the logged-in user is used.
is-meeting-host	N	Boolean value that, if <code>true</code> , indicates that the providers are user-configured and returns a list of providers based on the value of <code>principal-id</code> . The default value is <code>false</code> .

### Permission

For a user-configured provider, you must have view permissions on the provider. For account-level providers, you must have view permission on the account.

### Example

Request:

```
http://connectdev1/api/xml?action=telephony-provider-list&is-meeting-host=true&principal-id=11032
```

Response:

```
<results>
  <status code="ok"/>
  -
  <providers-user>
  -
    <provider provider-id="11722" acl-id="11032" provider-type="user-conf">
      <adaptor-id>11722-adaptor</adaptor-id>
      <name>Test</name>
      <provider-status>enabled</provider-status>
    </provider>
  </providers-user>
</results>
```

## telephony-provider-update

Creates a new provider or updates an existing provider. Only users who are administrators or meeting hosts can create a provider.

- To create or update a user-configured provider for the currently logged-in user, pass a value of `true` for `is-meeting-host`.
- To create or update a user-configured provider for a particular user, pass the user's principal ID to `principal-id` and pass a value of `true` for `is-meeting-host`.
- To update an account-level provider for a specified account, pass the account's ID to `account-id` and pass a value of `false` for `is-meeting-host`.

### Syntax

```
http://server_name/api/xml?action=telephony-provider-update&is-meeting-host=boolean&principal-id=integer&name=name
```

### Parameters

Name	Required?	Description
provider-id	Y if you are updating a provider; otherwise N	Integer that specifies the ID of the provider to be updated. If not specified, a new provider is created.
account-id	N	The ID of the account to which this provider belongs. If not specified, the ID of the logged-in account is used.
is-meeting-host	N	Boolean value that, if <code>true</code> , indicates that the provider is user-configured and creates or updates the provider based on the value of <code>principal-id</code> .

Name	Required?	Description
principal-id	N	Integer that specified the principal ID of the user for whom a provider needs to be created or updated. If not specified, the logged-in user's principal ID is used.
provider-status	N	Boolean value that specifies the status of the provider that is being updated or created. Acceptable values are <code>enabled</code> and <code>disabled</code> . If you are creating a new provider, the default value is <code>enabled</code> .
name	Y if you are creating a new provider; otherwise N	String that specifies the name of the provider being created or updated.

### Permission

If you are creating an account-level provider, you must have modify permission on the account. If you are creating a user-configured provider, you must be a member of the meeting hosts or administrators group. If you are updating a provider, you must have modify permission on the provider.

### Example

Request (for creating a provider):

```
http://connectdev1/api/xml?action=telephony-provider-update&is-meeting-host=true&principal-id=11032&name=my-provider2
```

Response:

```
<results>
  <status code="ok"/>
  -
  <telephony-provider provider-status="enabled" provider-type="user-conf" provider-id="12125">
    <name>my-provider2</name>
    <adaptor-id>12125-adaptor</adaptor-id>
  </telephony-provider>
</results>
```

Request (for updating a provider):

```
http://connectdev1/api/xml?action=telephony-provider-update&is-meeting-host=true&principal-id=11032&provider-id=11814&provider-status=disabled
```

Response:

```
<results>
  <status code="ok"/>
</results>
```

# Chapter 10: Common reference

This reference section describes XML elements and attributes that are used by more than one action in Adobe® Connect™ Web Services. The elements described here are referenced from the request and response tables describing actions in the Web Services XML API.

All parameter, element, and attribute names and values are case sensitive. For example, `name` is not the same as `Name`, and `sco-id` is not equivalent to `sco-ID`. You must enter them exactly as shown in this reference.

## Common XML elements and attributes

### access

#### Description

An attribute describing the level of access a user has to a course or curriculum.

#### Values

Value	Description
<code>access-blocked</code>	The course or curriculum is restricted and users cannot take it.
<code>access-hidden</code>	The course or curriculum is restricted, users cannot take it, and it is hidden in Adobe Connect Central (or the user interface of a custom application, if you use this value).
<code>access-open</code>	The course or curriculum is open and users can take it.
<code>access-optional</code>	The course or curriculum is optional.
<code>access-pass</code>	The user has already taken the course or curriculum and passed it.

### feature-id

#### Description

An attribute describing a feature that either users can use or things that can occur during a meeting. Use `feature-id` with the [meeting-feature-update](#) action.

For more information on the pods that can be enabled or disabled, see the *Adobe Connect User Guide*.

**Common reference**

**Values**

<b>Value</b>	<b>Description of functionality when value is enabled</b>
fid-archive	Lets a host start and stop the recording of a meeting. Disabling this setting means that recording settings are not controllable by the host.  To set Connect to automatically record all meetings, you must both disable <code>fid-archive</code> and enable <code>fid-archive-force</code> .
fid-archive-force	Sets all meetings to be recorded upon the start of the meeting. Recorded meetings appear in Adobe Connect Central.
fid-archive-publish-link	When meetings are set to be automatically recorded (by enabling <code>fid-archive-force</code> ), lets host create a link to the recording in the meeting folder.
fid-chat-transcripts	Creates a transcript file (one per meeting session) of all text messages exchanged in Chat, Q&A, and IM Pods.
fid-meeting-app-sharing	Lets host request control of attendee's input device (mouse or keyboard) when attendee is sharing their screen, desktop, or application.
fid-meeting-auto-promote	Lets hosts enable the option to automatically promote participants to presenters.
fid-meeting-breakout	Allows users to create breakout meetings.
fid-meeting-chat	Enables the Chat pod.
fid-meeting-chat-clear	Automatically clears Chat pod history when a new session of an existing meeting is started.
fid-meeting-chat-presenter	Lets users send chat messages only to the presenter.
fid-meeting-chat-private	Lets users send chat messages to specific attendees.
fid-meeting-chat-public	Lets users send chat messages to all attendees.
fid-meeting-desktop-sharing	Lets users share their screen (both the complete desktop and individual applications).
fid-meeting-dialout	Lets users use Call Out and Call Me features.
fid-meeting-disclaimer	Shows a disclaimer (for example, explaining that the meeting is being recorded) when a user starts or attends any meeting in this account. To proceed with the meeting, the host or attendees must first accept the disclaimer. If a user does not accept, the disclaimer user is returned to the Connect home page.  To set the text of the disclaimer, use the <a href="#">meeting-disclaimer-update</a> action.
fid-meeting-enhanced-rights	Lets host change the access of attendees to specific subfeatures.
fid-meeting-file-share	Enables the File Share pod.
fid-meeting-flv	Lets users use FLV files and mp3 files in the meeting.
fid-meeting-full-screen-affects-all	Enables "Presenter Changes affect everyone" for full-screen mode. Note: this feature does not enable or disable users' ability to enter full-screen mode.
fid-meeting-hold	Lets hosts place participants on hold.
fid-meeting-host-cursors	Lets a host change the display of the host's cursor.
fid-meeting-im	Enables the Instant Messages pod. This feature is part of the Adobe Connect integration with supported Microsoft real-time communication servers.  Disable this feature when you want to show the Invitee List pod ( <code>fid-meeting-invitee-presence=true</code> ) but hide the associated Instant Messages pod.
fid-meeting-invitee-presence	Enables the Invitee List pod and the associated Instant Messages pod. This feature is part of the Adobe Connect integration with supported Microsoft real-time communication servers.
fid-meeting-manage-link	Enables the "Manage Room with Web Manager" link in the meeting menu.

**Common reference**

<b>Value</b>	<b>Description of functionality when value is enabled</b>
fid-meeting-note	Enables the Notes pod.
fid-meeting-notes-clear	Automatically clears Notes pod history when a new session of an existing meeting is started.
fid-meeting-passcode-notallowed	Meeting hosts can assign passcodes to meetings in Connect Central. However, by default, this feature is disabled. Account Administrators can enable the feature in Connect Central or you can pass this feature ID and set <code>enable=false</code> .
fid-meeting-pause-annotate	Lets users pause during screen sharing and annotate on an overlay white board.
fid-meeting-people-list	Enables the Attendee List pod.
fid-meeting-poll	Enables the Poll pod.
fid-meeting-pres-only	Enables the use of the Presenter-Only area.
fid-meeting-pr-mode	Enables Preparing Mode, which lets hosts change the meeting layout without affecting other users' layouts. When the layout is ready, disable this mode to show the layout to all users.
fid-meeting-qa-clear	Automatically clears Q & A pod history when a new session of an existing meeting is started.
fid-meeting-questions	Enables the Q & A pod.
fid-meeting-room-bg	Lets a host change a meeting room background. Backgrounds are typically set in the meeting template.
fid-meetings-custom-pods	Enables custom pods.
fid-meeting-shared-library	Lets users select documents from the Content Library while in the Share pod. (This setting doesn't affect the File Share pod.)
fid-meeting-shared-upload	Lets users upload documents to the Content Library while in the Share pod. (This setting doesn't affect the File Share pod.)
fid-meeting-show-talker	Lets hosts enable or disable the "Who's Speaking" area in a meeting.
fid-meeting-show-talker-area	Enables the "Who's Speaking" area.
fid-meeting-video	Enables the Camera pod.
fid-meeting-voip	Enables voice controls within the meeting user interface, such as the ability to broadcast audio using VoIP.
fid-meeting-web-links	Enables the Web Links pod.
fid-meeting-white-board	Enables use of the white board in the Share pod.
fid-start-meeting-button	Lets a user start the meeting again after the host ends the meeting.
fid-training-openenroll	Enables open enrollment, which lets users enroll themselves in a course without approval from the Training Manager.  Disabling this feature for an account means that a training manager cannot change the settings of a course in Adobe Connect Central to let learners enroll without approval.
fid-archive-localdownload-notallowed	Enables or disables downloading of recordings locally.

**field**

**Description**

An element containing information about a custom field defined for an object or principal.



## Attributes

Attribute	Type	Description
permission-id	Allowed value	The permission needed to access the custom field. See <a href="#">permission-id</a> for valid values.
object-type	Allowed value	A definition for a valid object on the server (see <a href="#">type</a> for values).
field-id	String	A piece of text that identifies the custom field. Adobe Connect Central does not display the <code>field-id</code> , but various actions return it.
account-id	Integer	An ID for the user who is presently logged in, assigned by the server.
display-seq	Integer	The order in which the custom field is displayed in the user interface or returned by the action.
field-type	String	The type of data the field accepts. Allowed values are <code>text</code> , <code>textarea</code> , and <code>password</code> .
is-primary	Boolean	Whether the custom field can be deleted. <code>true</code> means the field cannot be deleted. <code>false</code> means it can.
is-required	Boolean	Whether the custom field is required. <code>true</code> means a value must be specified for this field in each principal or SCO that uses it. <code>false</code> means values for this field are not required.
acl-id	Integer	The custom field's ID in an access control list (ACL).
custom-seq	Integer	The sequence number assigned to the custom field in UI display.
type	String	The type of custom field (see <a href="#">type</a> for values).
principal-id	Integer	The ID of the principal for whom the custom field is defined.

## icon

### Description

The symbol used to identify a SCO in Connect Central.

### Values

Value	Description
archive	An archive of an Adobe Connect meeting.
attachment	A piece of content uploaded as an attachment.
authorware	A piece of multimedia content created with Macromedia® Authorware® from Adobe.
captivate	A demo or movie created with Adobe Captivate™.
course	A training course.
curriculum	A curriculum.
external-event	An external training that can be added to a curriculum.
flv	A media file in the FLV file format.
html	An HTML file.
image	An image.

**Common reference**

Value	Description
lms-plugin	A piece of content from an external learning management system.
logos	A custom logo used in a meeting room or Adobe Connect Central.
meeting-template	A custom look and feel for a meeting.
mp3	An MP3 file.
pdf	An Adobe Portable Document Format file.
pod	A visual box that provides functionality in a meeting room layout.
presentation	A presentation created with an earlier version of Adobe Breeze® software.
producer	A presentation created with Adobe Presenter.
seminar	A seminar created with Adobe Connect Seminars.
session	One occurrence of a recurring Adobe Connect meeting.
swf	A SWF file.

**lang****Description**

A two-letter or three-letter code describing a language according to the ISO 639 specifications. ISO 639-1 describes two-letter codes, and ISO 639-2 describes three-letter codes. The language code affects a Adobe Connect application display, for example, a meeting room or a Adobe Connect Central display.

**Values**

Two-Letter Value	Three-Letter Value	Language
en	eng	English
fr	fre	French (do not use fra)
de	ger	German (do not use deu)
ja	jpn	Japanese
ko	kor	Korean

**object-type****Description**

An attribute describing the type of a Adobe Connect object.

**Values**

Value	Meaning
object-type-account	An account that contains principals and SCOs.
object-type-action	An action in the Web Services XML API.
object-type-event	An Adobe Connect event.

**Common reference**

Value	Meaning
object-type-hidden	A SCO that is not visible in Adobe Connect Central (or in your application, if you use this value).
object-type-meeting	An Adobe Connect meeting.
object-type-principal	A user or group.
object-type-readonly	A setting indicating that Adobe Connect Central displays some data, but a user cannot set the data in Adobe Connect Central.
object-type-sco	A SCO representing a meeting, course, curriculum, piece of content, folder, or any other object on the server.

## path-type

### Description

The `path-type` attribute describes a type of learning path between two objects in a curriculum, for example, whether one must be completed as a prerequisite to the next.

### Values

Value	Meaning
completion-none	The current SCO is not a completion requirement for the curriculum.
completion-required	The current SCO is a completion requirement.
prereq-none	The current learning object has no prerequisites.
prereq-required	The current SCO has a prerequisite that must be completed.
prereq-hidden	The target learning object is required as a prerequisite. The current learning object is hidden until the target learning object is completed.
prereq-suggested	The current SCO has a prerequisite that is recommended, not required.
preass-blocked	The current SCO has a test-out. If the enrollee passes, this item is locked. If the enrollee fails, this item is available.
preass-none	The current SCO has no test-outs.
preass-hidden	The current SCO has a test-out. If the user passes, the current SCO can be hidden from the user. If the user fails, the current SCO is visible and the user can enroll.
preass-optional	The current SCO has a test-out. If the user passes, the current SCO is no longer required to complete the curriculum.

## permission-id

### Description

The `permission-id` parameter (or attribute) defines a permission. Depending on the context of the action or response, the permission might be one a principal has on a SCO, or a permission that is needed in order to execute an action.

## Values

Permission	Description
view	The principal can view, but cannot modify, the SCO. The principal can take a course, attend a meeting as participant, or view a folder's content.
host	<b>Available for meetings only.</b> The principal is host of a meeting and can create the meeting or act as presenter, even without <code>view</code> permission on the meeting's parent folder.
mini-host	<b>Available for meetings only.</b> The principal is presenter of a meeting and can present content, share a screen, send text messages, moderate questions, create text notes, broadcast audio and video, and push content from web links.
remove	<b>Available for meetings only.</b> The principal does not have participant, presenter or host permission to attend the meeting. If a user is already attending a live meeting, the user is not removed from the meeting until the session times out.
publish	<b>Available for SCOs other than meetings.</b> The principal can publish or update the SCO. The <code>publish</code> permission includes <code>view</code> and allows the principal to view reports related to the SCO. On a folder, <code>publish</code> does not allow the principal to create new subfolders or set permissions.
manage	<b>Available for SCOs other than meetings or courses.</b> The principal can view, delete, move, edit, or set permissions on the SCO. On a folder, the principal can create subfolders or view reports on folder content.
denied	<b>Available for SCOs other than meetings.</b> The principal cannot view, access, or manage the SCO.

## Special permissions

The server defines a special principal, `public-access`, which combines with values of `permission-id` to create special access permissions to meetings:

- `principal-id=public-access` and `permission-id=view-hidden` means the Adobe Connect meeting is public, and anyone who has the URL for the meeting can enter the room.
- `principal-id=public-access` and `permission-id=remove` means the meeting is protected, and only registered users and accepted guests can enter the room.
- `principal-id=public-access` and `permission-id=denied` means the meeting is private, and only registered users and participants can enter the room.

## quota-ID

### Description

The `quota-ID` parameter defines a quota in the system. The quota type you specify determines the value of `acl-id` to use.

### Values

Quota type	Quota description	Corresponding acl-id to use
<code>live-user-quota</code>	The number of account-wide Meeting Attendees.	The account-id of the account.
<code>concurrent-user-per-meeting-quota</code>	The number of concurrent users in one meeting.	The tree-id of the user-meetings tree or meetings tree.
<code>training-user</code>	The number of concurrent Learners for one account.	The account-id of the account.
<code>num-of-members-quota</code>	The number of Authors or Meeting Hosts.	The principal-id for either the Authors group or the Meeting Hosts (live-admins), depending on which quota you want to specify.

## status

### Description

A status code returned by Adobe Connect in the response to all actions in the Web Services XML API.

### Response structure

```
<?xml version="1.0" encoding="utf-8" ?>  
<results>  
  <status code=allowedValue >  
</results>
```

or

```
<?xml version="1.0" encoding="utf-8" ?>  
<results>  
  <status code=allowedValue>  
    <invalid field=string type=allowedValue subcode=allowedValue />  
  </status>  
</results>
```

### Attributes

**code** The status of the response.

Value	Description
invalid	Indicates that a call is invalid in some way. The <code>invalid</code> element provides more detail.
no-access	Indicates that you don't have permission to call the action. The <code>subcode</code> attribute provides more details.
no-data	Indicates that there is no data available (in response to an action that would normally result in returning data). Usually indicates that there is no item with the ID you specified. To resolve the error, change the specified ID to that of an item that exists.
ok	Indicates that the action has completed successfully.
too-much-data	Indicates that the action should have returned a single result but is actually returning multiple results. For example, if there are multiple users with the same user name and password, and you call the <code>login</code> action using that user name and password as parameters, the system cannot determine which user to log you in as, so it returns a <code>too-much-data</code> error.

**subcode** If present, provides more detail describing the status of the response. For example, `subcode` values are used to differentiate between different situations in which `code` is set to `noaccess-`.

Value	Description
account-expired	The customer account has expired.
denied	Based on the supplied credentials, you don't have permission to call the action.
no-login	The user is not logged in. To resolve the error, log in (using the <code>login</code> action) before you make the call. For more information, see <a href="#">login</a> .
no-quota	The account limits have been reached or exceeded.
not-available	The required resource is unavailable.
not-secure	You must use SSL to call this action.
pending-activation	The account is not yet activated.

**Common reference**

Value	Description
pending-license	The account's license agreement has not been settled.
sco-expired	The course or tracking content has expired.
sco-not-started	The meeting or course has not started.

**The invalid element**

An element that gives information describing a status code of `invalid`.

Element	Attribute	Type	Description
invalid		Empty, with attributes	Information about why the call was invalid.
	field	String	The name of the request parameter that was incorrect or missing.
	type	Allowed value	The type of the incorrect or missing field.
	subcode	Allowed value	A code explaining why the request was invalid (see the following table for values).

The following table shows the allowed values for `subcode` when `code` is `invalid`.

Value	Description
duplicate	The call attempted to add a duplicate item in a context where uniqueness is required.
format	A passed parameter had the wrong format.
illegal-operation	The requested operation violates integrity rules (for example, moving a folder into itself).
missing	A required parameter is missing.
no-such-item	The requested information does not exist.
range	The value is outside the permitted range of values.

**Returned by**

All actions in the Adobe Connect Web Services XML API.

**Samples**

This is a successful response:

```
<status code="ok" />
```

This is an invalid response:

```
<status code="invalid">
<invalid field="principal-id" type="id" subcode="missing" />
</status>
```

**status attribute**

**Description**

An attribute that describes a user's progress with a course or curriculum SCO. It is returned by actions that provide training reports.

A curriculum or folder can only have a status of `completed` or `incomplete`.

The following table shows the allowed values for `status` when returned in a `row` element describing a course.

Value	Description
<code>user-passed</code>	The SCO has scored interactions the user has passed.
<code>user-failed</code>	The SCO has scored interactions. The user has answered them, but failed.
<code>completed</code>	The user has viewed all of the SCO's content, but the content has no scored interactions.
<code>incomplete</code>	The user has not viewed all of the SCO's content.
<code>not-attempted</code>	The user has not started viewing all of the SCO's content.
<code>review</code>	The user has passed or completed the course or used all available retries.

### Sample

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <report-user-training-transcripts>
    <row transcript-id="2006905612" sco-id="2006298431"
      principal-id="2006258745" status="review" score="0" max-score=""
      certificate="" type="content" icon="course">
      <name>Test Course</name>
      <url-path>/test/</url-path>
      <login>joy@acme.com</login>
      <date-taken>2006-06-30T15:23:55.070-07:00</date-taken>
      <principal-name>Joy Smith</principal-name>
    </row>
  </report-user-training-transcripts>
</results>
```

## time-zone-id

### Description

Settings that describe time zones that you can use with `time-zone-id`.

### Values

Time zone setting	Value
(GMT-12:00) International Date Line West	0
(GMT-11:00) Midway Island, Samoa	1
(GMT-10:00) Hawaii	2
(GMT-09:00) Alaska	3
(GMT-08:00) Pacific Time (US and Canada); Tijuana	4
(GMT-07:00) Mountain Time (US and Canada)	10
(GMT-07:00) Chihuahua, La Paz, Mazatlan	13
(GMT-07:00) Arizona	15

Time zone setting	Value
(GMT-06:00) Central Time (US and Canada)	20
(GMT-06:00) Saskatchewan	25
(GMT-06:00) Guadalajara, Mexico City, Monterrey	30
(GMT-06:00) Central America	33
(GMT-05:00) Eastern Time (US and Canada)	35
(GMT-05:00) Indiana (East)	40
(GMT-05:00) Bogota, Lima, Quito	45
(GMT-04:00) Atlantic Time (Canada)	50
(GMT-04:00) Caracas, La Paz	55
(GMT-04:00) Santiago	56
(GMT-03:30) Newfoundland	60
(GMT-03:00) Brasilia	65
(GMT-03:00) Buenos Aires, Georgetown	70
(GMT-03:00) Greenland	73
(GMT-02:00) Mid-Atlantic	75
(GMT-01:00) Azores	80
(GMT-01:00) Cape Verde Islands	83
(GMT) Greenwich Mean Time: Dublin, Edinburgh, Lisbon, London	85
(GMT) Casablanca, Monrovia	90
(GMT+01:00) Belgrade, Bratislava, Budapest, Ljubljana, Prague	95
(GMT+01:00) Sarajevo, Skopje, Warsaw, Zagreb	100
(GMT+01:00) Brussels, Copenhagen, Madrid, Paris	105
(GMT+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna	110
(GMT+01:00) West Central Africa	113
(GMT+02:00) Bucharest	115
(GMT+02:00) Cairo	120
(GMT+02:00) Helsinki, Kiev, Riga, Sofia, Tallinn, Vilnius	125
(GMT+02:00) Athens, Istanbul, Minsk	130
(GMT+02:00) Jerusalem	135
(GMT+02:00) Harare, Pretoria	140
(GMT+03:00) Moscow, St. Petersburg, Volgograd	145
(GMT+03:00) Kuwait, Riyadh	150
(GMT+03:00) Nairobi	155
(GMT+03:00) Baghdad	158



**Common reference**

<b>Time zone setting</b>	<b>Value</b>
(GMT+03:30) Tehran	160
(GMT+04:00) Abu Dhabi, Muscat	165
(GMT+04:00) Baku, Tbilisi, Yerevan	170
(GMT+04:30) Kabul	175
(GMT+05:00) Ekaterinburg	180
(GMT+05:00) Islamabad, Karachi, Tashkent	185
(GMT+05:30) Chennai, Kolkata, Mumbai, New Delhi	190
(GMT+05:45) Kathmandu	193
(GMT+06:00) Astana, Dhaka	195
(GMT+06:00) Sri Jayawardenepura	200
(GMT+06:00) Almaty, Novosibirsk	201
(GMT+06:30) Rangoon	203
(GMT+07:00) Bangkok, Hanoi, Jakarta	205
(GMT+07:00) Krasnoyarsk	207
(GMT+08:00) Beijing, Chongqing, Hong Kong SAR, Urumqi	210
(GMT+08:00) Kuala Lumpur, Singapore	215
(GMT+08:00) Taipei	220
(GMT+08:00) Perth	225
(GMT+08:00) Irkutsk, Ulaan Bataar	227
(GMT+09:00) Seoul	230
(GMT+09:00) Osaka, Sapporo, Tokyo	235
(GMT+09:00) Yakutsk	240
(GMT+09:30) Darwin	245
(GMT+09:30) Adelaide	250
(GMT+10:00) Canberra, Melbourne, Sydney	255
(GMT+10:00) Brisbane	260
(GMT+10:00) Hobart	265
(GMT+10:00) Vladivostok	270
(GMT+10:00) Guam, Port Moresby	275
(GMT+11:00) Magadan, Solomon Islands, New Caledonia	280
(GMT+12:00) Fiji Islands, Kamchatka, Marshall Islands	285
(GMT+12:00) Auckland, Wellington	290
(GMT+13:00) Nuku'alofa	300

## type

### Description

A return element or attribute defining the type of a SCO or principal on the server. The allowed values for `type` are different for SCOs and principals.

### SCO types

A SCO can be content, a meeting, an event, a curriculum, a folder or tree, or other object on Adobe Connect. Most SCOs can have any of the following values for `type`:

Value	Description
content	A viewable file uploaded to the server, for example, an FLV file, an HTML file, an image, a pod, and so on.
curriculum	A curriculum.
event	A event.
folder	A folder on the server's hard disk that contains content.
link	A reference to another SCO. These links are used by curriculums to link to other SCOs. When content is added to a curriculum, a link is created from the curriculum to the content.
meeting	An Adobe Connect meeting.
session	One occurrence of a recurring Adobe Connect meeting.
tree	The root of a folder hierarchy. A tree's root is treated as an independent hierarchy; you can't determine the parent folder of a tree from inside the tree.

Content objects returned by some actions (for example, `report-bulk-objects`) have the `type` values shown in the following table:

Value	Description
archive	An archived copy of a live Adobe Connect meeting or presentation.
attachment	A piece of content uploaded as an attachment.
authorware	A piece of multimedia content created with Macromedia Authorware from Adobe.
captivate	A demo or movie authored in Adobe Captivate.
curriculum	A curriculum, including courses, presentations, and other content.
external-event	An external training that can be added to a curriculum.
flv	A media file in the FLV file format.
image	An image, for example, in GIF or JPEG format.
meeting	An Adobe Connect meeting.
presentation	A presentation.
swf	A SWF file.

### Principal types

For principals, the allowed values for `type` are shown in the following table:

**Common reference**

<b>Value</b>	<b>Description</b>
admins	The built-in group Administrators, for Adobe Connect server Administrators.
authors	The built-in group Authors, for authors.
course-admins	The built-in group Training Managers, for training managers.
event-admins	The built-in group Event Managers, for anyone who can create an Adobe Connect meeting.
event-group	The group of users invited to an event.
everyone	All Adobe Connect users.
external-group	A group authenticated from an external network.
external-user	A user authenticated from an external network.
group	A group that a user or Administrator creates.
guest	A non-registered user who enters an Adobe Connect meeting room.
learners	The built-in group learners, for users who take courses.
live-admins	The built-in group Meeting Hosts, for Adobe Connect meeting hosts.
seminar-admins	The built-in group Seminar Hosts, for seminar hosts.
user	A registered user on the server.

**Custom field types**

When used with a custom field, `type` can have any of the following values.

<b>Value</b>	<b>Description</b>
required	A required custom field for the account.
optional	An optional field that is displayed during self-registration.
optional-no-self-reg	An optional field that is hidden during self-registration.

# Chapter 11: Sample application

To understand how to use Adobe Connect Web Services to build a custom application, download the sample application from [www.adobe.com/go/learn\\_cnn\\_firstapp\\_en](http://www.adobe.com/go/learn_cnn_firstapp_en).

## Getting started with the sample application

To understand how to use Adobe Connect Web Services to build a custom application, download the sample application from [www.adobe.com/go/learn\\_cnn\\_firstapp\\_en](http://www.adobe.com/go/learn_cnn_firstapp_en).

The sample application is written in Java and JSP using a model-view-controller architecture and runs on any web application server with a J2EE servlet container. The sample demonstrates how to implement Adobe Connect meeting functionality in a Java custom application or portal, showing how to log in a user, list a user's meetings, and create, update, and delete meetings.

As you build and design your application, there are several points about the Adobe Connect Web Services XML API you should keep in mind:

**Sequence of API calls** Calls to the XML API often need to be made in a specific sequence. For example, you need to get the `principal-id` of a user and the `sco-id` of a meeting before you call `permissions-update` to make the user a meeting presenter. Call sequences for various tasks are described in the first chapters of this guide.

**Different parameter names for the same value** A value might have one parameter name in one call and a different parameter name in another call. For example, the unique ID of a SCO might be a `sco-id` when used with `sco-info`, but an `acl-id` in `permissions-update`. It's the same value in both calls. The best way to learn this is to use the API reference in this guide.

**SCOs are not object-oriented** A SCO is a shareable content object on the server (for a complete definition, see [Find SCOs](#)). A SCO can be a meeting, presentation, course, image, folder, or any object on the server. SCOs are stored within folders in a navigation hierarchy. However, there is no object-oriented structure for SCOs, and one type of SCO is not a subclass of another type. Keep this in mind as you design your application.

## Build an adapter class

*Note:* See the sample files `XMLApiAdapter.java`, `login.jsp`, and `mymeetings.jsp`

When you are building a custom application, it's very handy to have an adapter class. You create an instance of the class for each user login session, and the adapter handles connecting to the server, logging the user in, making requests to the XML API, and parsing XML responses.

### Write constructors for the adapter class

The following constructor (from the sample application file `XMLApiAdapter.java`) creates an instance of the adapter class to represent a user accessing Adobe Connect. This is the constructor to use when you already have the `BREEZESSESSION` cookie value (see [Log in from an application](#)). The constructor also calls the `createXPaths` method to create valid XPath instances to use in other methods:

**Sample application**

```
public XMLApiAdapter(String baseUrl, String breezesession)
    throws XMLApiException {
    this.setBaseUrl(baseUrl);
    this.breezesession = breezesession;
    createXPath();
}

```

The second constructor takes a user's login ID and password, as well as a BREEZESSESSION cookie value:

```
public XMLApiAdapter(String baseUrl, String login, String password,
    String breezesession) throws XMLApiException {
    this(baseUrl, breezesession);
    this.setLogin(login);
    this.setPassword(password);
}

```

You can get the BREEZESSESSION cookie value before the user logs in by calling `common-info`.

**Create an instance of the adapter**

The following code (from `mymeetings.jsp`) creates an instance of the `XMLApiAdapter` class to represent a user who is logged in to Adobe Connect. The current value of `breezesession`, which holds the BREEZESSESSION cookie value, is then stored in the JSP `session` attribute for other files to access.

```
<%! XMLApiAdapter breeze = null; %>

<%
...
breeze = new XMLApiAdapter(breezeBase, login, password, breezesession);
breeze.getBreezesession();
session.setAttribute("breezesession", breeze);
...
%>

```

**Log the user in**

**Note:** See the sample files `XMLApiAdapter.java` and `login.jsp`.

Your application needs a method that logs users in to Adobe Connect. A login method needs to open a connection to the server, call the `login` action, and get the XML response. The method also needs to read the value of the BREEZESSESSION cookie from the response header and store the value.

The simplest form of the `login` action is:

```
https://example.com/api/xml?action=login&login=joy@example.com
    &password=jazz

```

You might also need to add `session`, `account-id`, or `domain` parameters to the `login` action (see [Log in from an application](#) for more ways to call `login`).

A successful login returns this response, with a status code of `ok`:

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
</results>

```

## Build the base request URL

The login method should first build the base request URL to send to the server. In the sample, the `breezeUrl` method builds a URL like this:

```
http://example.com/api/xml?action=
```

The method also adds an action name and query string that you pass to it. This is the full method:

```
protected URL breezeUrl(String action, String queryString)
    throws MalformedURLException {
    return new URL(getBaseUrl() + "/api/xml?" + "action=" + action
        + (queryString != null ? ('&' + queryString) : ""));
}
```

## Send the user's login information

The `login` method calls the `login` action, opens the connection to the server, reads the `BREEZESSESSION` cookie from the response header, and then checks for a status code of `ok` in the response:

```
protected void login() throws XMLApiException {
    try {
        if (breezesession != null)
            logout();

        URL loginUrl = breezeUrl("login", "login=" + getLogin()
            + "&password=" + getPassword());

        URLConnection conn = loginUrl.openConnection();
        conn.connect();

        InputStream resultStream = conn.getInputStream();
        Document doc = new SAXBuilder(false).build(resultStream);

        String breezesessionString = (String) (conn
            .getHeaderField("Set-Cookie"));

        StringTokenizer st = new StringTokenizer(breezesessionString, "=");
        String sessionName = null;
    }
}
```

```
        if (st.countTokens() > 1)
            sessionName = st.nextToken();

        if (sessionName != null&&
            sessionName.equals("BREEZESESSION")) {
            String breezesessionNext = st.nextToken();
            int semiIndex = breezesessionNext.indexOf(';');
            breezesession = breezesessionNext.substring(0, semiIndex);
        }

        Element root = doc.getRootElement();
        String status = getStatus(root);
        if (breezesession == null || !"ok".equalsIgnoreCase(status))
            throw new XMLApiException("Could not log into Adobe Connect.");
    } catch (IOException ioe) {
        throw new XMLApiException(IO_ERROR, ioe);
    } catch (JDOMException jde) {
        throw new XMLApiException(PARSE_ERROR, jde);
    }
}
```

## Send XML requests

*Note: See the sample files XMLApiAdapter.java and createmeeting.jsp.*

Once a user is logged in, it's useful to have a generic request method that sends a request to the server when you provide an action name and query string.

The request method in the sample takes an action and a query string and sends the BREEZESESSION cookie value back to the server in the request header:

```
protected Element request(String action, String queryString)
    throws XMLApiException {
    try {
        if (breezesession == null)
            login();

        URL url = breezeUrl(action, queryString);

        URLConnection conn = url.openConnection();
        conn.setRequestProperty("Cookie", "BREEZESESSION=" + breezesession);
        conn.connect();

        InputStream resultStream = conn.getInputStream();
        Document doc = new SAXBuilder(false).build(resultStream);
        return doc.getRootElement();

    } catch (IOException ioe) {
        throw new XMLApiException("A communication error occurred", ioe);
    } catch (JDOMException jde) {
        throw new XMLApiException("A parsing error occurred", jde);
    }
}
```

## Parse XML responses

*Note:* See the sample file `XMLApiAdapter.java`.

When you send an XML request to Adobe Connect, the server returns an XML response. You need to parse the response and extract values, including status codes. One way to parse the response is to use XPath to traverse XML elements (see the XPath tutorial at [w3schools.com](http://w3schools.com) for more information).

As an example, this is the response from `sco-shortcuts`:

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <shortcuts>
    <sco tree-id="4930295" sco-id="2006258748" type="my-courses">
      <domain-name>http://example.com</domain-name>
    </sco>
    <sco tree-id="4930293" sco-id="2006258749" type="my-events">
      <domain-name>http://example.com</domain-name>
    </sco>
    ...
  </shortcuts>
</results>
```

### Extract values

The `getShortcuts` method calls `sco-shortcuts` and parses the response using XPath. This is an example of how to extract a list of `sco` elements and the `sco-id` of each:

```
public List getShortcuts() throws XMLApiException {
    try {
        Element e = request("sco-shortcuts", null);
        List scosXml = XPath.selectNodes(e, "//sco");
        List scos = new ArrayList();
        XPath id = XPath.newInstance("./@sco-id");
        for (Iterator i = scosXml.iterator(); i.hasNext();) {
            Element s = (Element) i.next();
            SCO sco = getSco(id.valueOf(s));
            if (null != sco)
                scos.add(sco);
        }
        return scos;
    } catch (JDOMException jde) {
        throw new XMLApiException(PARSE_ERROR, jde);
    }
}
```

### Extract a status code

Your application also needs to parse both successful and unsuccessful responses for status codes. For example, when a user calls an action without sufficient permission, the error response has a `status` element with both `code` and `subcode` attributes:



**Sample application**

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="no-access" subcode="denied" />
</results>
```

These lines in the `createXPath`s method parse for the values of `code` and `subcode`:

```
codePath = XPath.newInstance("//status/@code");
subcodePath = XPath.newInstance("//status/@subcode");
```

In the sample, the `createXPath`s method is called when you create an instance of `XMLApiAdapter`. The `getStatus` method then uses `codePath` and `subcodePath` to return the code and subcode values, separated by a hyphen:

```
private String getStatus(Element e1) throws JDOMException {
    String code = codePath.valueOf(e1);
    String subcode = subcodePath.valueOf(e1);
    StringBuffer status = new StringBuffer();
    if (null != code && code.length() > 0)
        status.append(code);
    if (null != subcode && subcode.length() > 0)
        status.append(" - " + subcode);
    return status.toString();
}
```

## Display user information

*Note:* See the sample files `XMLApiAdapter.java`, `UserInfo.java`, and `header.jsp`.

In your user interface, you might want to display information about a user, such as a name, during the user's login session.

You can retrieve simple information about the user by calling `common-info` after the user logs in, like this:

```
https://example.com/api/xml?action=common-info
```

The response has a `user` element with information you can display or store in variables to use later:

```
<user user-id="2006258745" type="user">
  <name>Joy Smith</name>
  <login>joy@acme.com</login>
</user>
```

If you call `common-info` before the user logs in, the response does not have a `user` element.

## Get information about the user

In the sample, the `getUserInfo` method calls `common-info` and parses the response for the values of `name`, `login`, and `user-id`. The method then stores information about the user in an instance of the `UserInfo` class, which is a standard bean class with getter and setter methods.

**Sample application**

```
public UserInfo getUserInfo(String login, String password)
    throws XMLApiException {
    try {
        Element e = request("common-info", "login=" + login + "&password="
            + password);
        XPath name = XPath.newInstance("//user/name");
        XPath log = XPath.newInstance("//user/login");
        XPath id = XPath.newInstance("//user/@user-id");

        UserInfo user = new UserInfo();
        user.setLogin(log.valueOf(e));
        user.setName(name.valueOf(e));
        user.setUserId(id.valueOf(e));

        return user;

    } catch (JDOMException jde) {
        throw new XMLApiException(PARSE_ERROR, jde);
    }
}
```






## List a user’s meetings

You may want to list a user’s meetings in your application. You can choose the meetings to list and the information to display based on your application design. This illustration shows one example of a meeting list:

**Scheduled Meetings**

Name ▶	Start Time ▶	Role ▶
 <a href="#">October Company Meeting</a> <input type="button" value="Enter"/>	10/01/2006 9:00 AM	Host

**Expired meetings**

Name ▶	Start Time ▶	Role ▶
 <a href="#">EN - Monday Night Football</a> <input type="button" value="Enter"/>	05/17/2004 3:30 PM	Participant
 <a href="#">seminar invitee</a> <input type="button" value="Enter"/>	02/13/2005 12:00 AM	Host
 <a href="#">How to Write a Novel</a> <input type="button" value="Enter"/>	09/06/2006 10:00 PM	Presenter
 <a href="#">September All Hands Meeting</a> <input type="button" value="Enter"/>	09/15/2006 9:00 AM	Host
 <a href="#">Platinum Support Team Meeting</a> <input type="button" value="Enter"/>	08/02/2006 11:45 AM	Participant

To list a user’s meetings using the XML API, call `report-my-meetings` with or without a filter. Without a filter, `report-my-meetings` returns all of a user’s meetings:

```
https://example.com/api/xml?action=report-my-meetings
```

You can add `filter-expired=false` to return only meetings that are currently in progress and scheduled in the future:

```
https://example.com/api/xml?action=report-my-meetings&filter-expired=false
```

Even with a filter, the response is likely to have multiple `meeting` elements that you need to iterate through and extract data from. A `meeting` element looks like this:

```
<meeting sco-id="2007063179" type="meeting" icon="meeting"
    permission-id="host" active-participants="0">
  <name>September All Hands Meeting</name>
  <description>For all company employees</description>
  <domain-name>example.com</domain-name>
  <url-path>/sept15/</url-path>
  <date-begin>2006-09-15T09:00:00.000-07:00</date-begin>
  <date-end>2006-09-15T18:00:00.000-07:00</date-end>
  <expired>false</expired>
  <duration>09:00:00.000</duration>
</meeting>
```

## Get the meeting list

To get the meeting list in Java, write a method like `getMyMeetings`, which lists a user's meetings with a filter as an argument. If you don't want to use a filter, you can pass `null` as the filter argument. A meeting is a SCO, so `getMyMeetings` calls the `getSco` method to extract values from the response and store them in an instance of `SCO.java`.

```
public List getMyMeetings(String filter) throws XMLApiException {
    try {
        Element meetingDoc = request("report-my-meetings", filter);
        List list = XPath.selectNodes(meetingDoc, "//meeting");
        Iterator meetings = list.iterator();
        List result = new ArrayList();
        while (meetings.hasNext()) {
            Element m = (Element) meetings.next();
            SCO meeting = getSco(m);
            result.add(meeting);
        }
        return result;
    } catch (JDOMException jde) {
        throw new XMLApiException(PARSE_ERROR, jde);
    }
}
```

The `SCO` object encapsulates data about the SCO so you can easily retrieve it from a web page (for example, HTML or JSP) in your application.

## Create and update meetings

*Note:* See the sample files `XMLApiAdapter.java` and `SCO.java`.

You might also want to allow users to create meetings in your application. To create a meeting, call `sco-update` with the `folder-id` of a meetings folder and `type=meeting`:

```
https://example.com/api/xml?action=sco-update
    &folder-id=2006258750&description=For all company employees
    &name=Company All Hands Meeting&type=meeting&lang=en
    &date-begin=2006-06-16T23:00&date-end=2006-06-16T23:30
```

The response returns the `sco-id` of the meeting, which you can extract and store:

**Sample application**

```
<sco account-id="624520" disabled="" display-seq="0" folder-id="2006258750" icon="meeting"
lang="en" max-retries="" sco-id="2006743452"
  source-sco-id="-1625529" type="meeting" version="1">
```

The difference between calling `sco-update` to create or update a meeting is:

- Pass a `folder-id` to create a meeting.
- Pass a `sco-id` to update an existing meeting. A meeting only has a `sco-id` if it already exists.

## Create a meeting

In an application, you first need to collect from the user the information needed to create or update the meeting, such as the meeting name, date, time, and so on. With that information, use a method such as `updateSCO` that calls the `sco-update` action.

In `sco-update`, be sure to set the `type` of the SCO to `meeting`. As an option, you can also set a language code for the meeting room, such as `lang=en`, for example:

```
https://example.com/api/xml?action=sco-update&folder-id=2006258750
  &description=nov&name=Nov%20All%20Hands%20Meeting&type=meeting&lang=en
  &date-begin=2006-11-11T09:00&date-end=2006-11-11T17:00
```

The `updateSCO` method shows how to implement the `sco-update` call in Java, once you collect information about the meeting from the user:

**Sample application**

```

public String updateSCO(String action, SCO sco) throws XMLApiException {
    try {
        StringBuffer sb = new StringBuffer();
        Map data = sco.getUpdateFields();

        if (CREATE.equals(action))
            sb.append("folder-id=" + sco.getFolderId());
        else
            sb.append("sco-id=" + sco.getId());
        Iterator iter = data.keySet().iterator();
        while (iter.hasNext()) {
            String key = (String) iter.next();
            if (key.indexOf("sco-id") != -1)
                continue;
            if (key.indexOf("folder-id") != -1)
                continue;
            String value = (String) data.get(key);
            sb.append("&" + key + "=" + value);
        }
        if (null == data.get("type"))
            throw new XMLApiException("SCO type not defined");
        Element e = request("sco-update", sb.toString());
        XPath scoId = XPath.newInstance("//results/sco/@sco-id");
        if (scoId.valueOf(e) == null)
            return null;
        else
            return scoId.valueOf(e);
    } catch (JDOMException jde) {
        throw new XMLApiException(PARSE_ERROR, jde);
    } catch (ParseException pe) {
        throw new XMLApiException(PARSE_ERROR, pe);
    }
}

```

**Set meeting access**

Once you have a `sco-id`, a meeting needs access. The user who creates the meeting is the host by default and chooses whether the meeting is public or private, set by a combination of `permission-id` and `principal-id` in [permissions-update](#). For example, this call makes a meeting public:

```

https://example.com/api/xml?action=permissions-update&acl-id=2006334033
&principal-id=public-access&permission-id=view-hidden

```

Once a user chooses these values, the `setPermissions` method calls `permissions-update` to set the meeting access:

```

public void setPermissions(String aclId, String principalId,
    String permissionId) throws XMLApiException {
    request("permissions-update", "acl-id=" + aclId + "&principal-id="
        + principalId + "&permission-id=" + permissionId);
}

```

**Display meeting detail**

*Note:* See the sample files `XMLApiAdapter.java`, `SCO.java`, `mymeetings.jsp`, and `showmeeting.jsp`.

Most of the information you want to display about a meeting comes from `sco-info`:

```
https://example.com/api/xml?action=sco-info&sco-id=2006334909
```

The response has many values that you can display, for example:

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <sco account-id="624520" disabled="" display-seq="0"
        folder-id="2006258747" icon="producer" lang="en" max-retries=""
        sco-id="2006334909" source-sco-id="" type="content" version="1">
    <date-created>2006-05-11T12:00:02.000-07:00</date-created>
    <date-modified>2006-05-16T15:22:25.703-07:00</date-modified>
    <name>Test Quiz</name>
    <url-path>/quiz</url-path>
    <passing-score>10</passing-score>
    <duration>15100.0</duration>
    <section-count>6</section-count>
  </sco>
</results>
```

## Get information about a SCO

The `getSco` Java method makes the call to `sco-info` and parses the result, storing values in variables so that you can display them in a user interface:

```
public SCO getSco(String scoId) throws XMLApiException {
    try {
        Element e = scoInfo(scoId);
        if(!"ok".equalsIgnoreCase(codePath.valueOf(e)))
            return null;
        Element sco = (Element) XPath.selectSingleNode(e, "//sco");
        ...
    }
}
```

## Construct the URL to the meeting room

You also need to create the URL to the meeting room. You can do this with a call to `sco-info` and another to `sco-shortcuts`:

```
https://example.com/api/xml?action=sco-info&sco-id=2006258750
```

```
https://example.com/api/xml?action=sco-shortcuts
```

Extract the `url-path` from the `sco-info` response. Then, extract the `domain-name` from the `sco-shortcuts` response and concatenate the two values:

```
<?xml version="1.0" encoding="utf-8" ?>
<results>
  <status code="ok" />
  <shortcuts>
    <sco tree-id="4930295" sco-id="2006258748" type="my-courses">
      <domain-name>http://example.com</domain-name>
    </sco>
  ..
</results>
```

You can also use a single call to `report-my-meetings` if the user is logged in and the meeting is in the user's `my-meetings` folder:

```
https://example.com/api/xml?action=report-my-meetings
```

**Sample application**

In this case, extract both the domain-name and url-path from the report-my-meetings response.

The `scoUrl` Java method constructs the URL by calling `sco-info` to retrieve the url-path and then `sco-shortcuts` to retrieve the domain-name. In this case, two calls are used so that you do not need to make the assumption that the meeting is in the current user's my-meetings folder:

```
public String scoUrl(String scoId) throws XMLApiException {
    try {
        Element e = request("sco-info", "sco-id=" + scoId);
        if(!(codePath.valueOf(e).equalsIgnoreCase("ok")))
            return "";
        XPath xpath = XPath.newInstance("//url-path/text()");
        String path = ((Text) xpath.selectSingleNode(e)).getText();

        e = request("sco-shortcuts", null);
        xpath = XPath.newInstance("//domain-name/text()");
        String url = ((Text) xpath.selectSingleNode(e)).getText();

        return url + "/" + path.substring(1) + "?session=" + breezeSession;
    } catch (JDOMException jde) {
        throw new XMLApiException(PARSE_ERROR, jde);
    }
}
```