



Apple Education

iTunes U Administrator's Guide

Chapter 2

May 15, 2006

Apple Computer, Inc.

Chapter 2

Integrating Authentication and Authorization Services

This chapter describes how to integrate iTunes U with your institution's existing authentication and authorization services.

iTunes U assumes that your institution is using an existing web-based infrastructure to manage your student, instructor, and other user IDs and that you can authenticate and authorize these users to iTunes U.

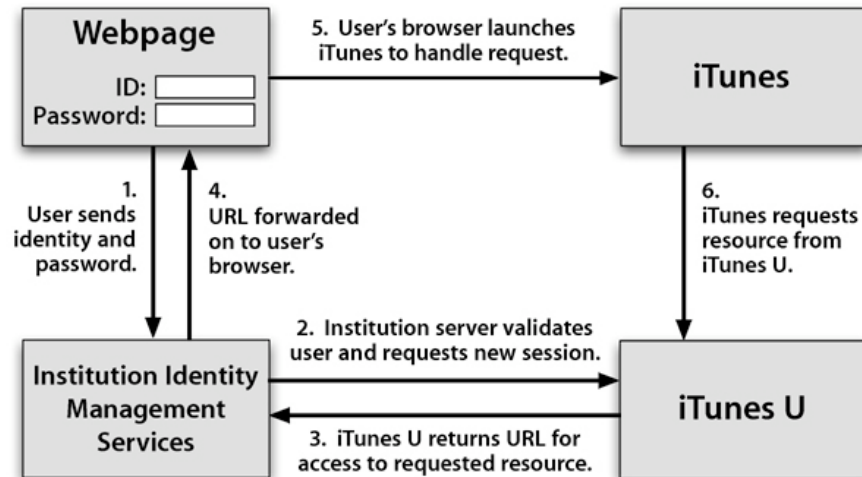
By using your existing services to validate user information and access privileges, you can ensure secure access to an iTunes U site containing course materials. Whether your site will be for your students, instructors, and staff only, or will also be available to the public, you must, at a minimum, have an existing web-based infrastructure in place to authenticate and authorize a site administrator. By using your systems to authenticate and authorize users, you maintain control over the user information sent to iTunes U.

This chapter covers the steps needed to integrate iTunes U with your authentication and authorization services:

- 1 Create local server scripts that detect user authentication and transfer the necessary authorization data to iTunes U when a user tries to access an iTunes U page from your webpages. For more information, see "Transferring Authorization Data to iTunes U" later in this document.
- 2 Determine where access to iTunes U should be added to your existing webpages, or consider developing new pages. For more information, see "Linking to iTunes U" later in this document.
- 3 Test your authentication and authorization integration using the iTunes U API. For more information, see "Testing Your iTunes U Integration Code" later in this document.

Once you have your scripts set up, you can test whether iTunes U understands the authorization data your system sends by transferring your site administrator's credentials to iTunes U as a token data string. The credentials you send, consisting of the user's attributes and unique course identifier, officially identify the user and confirm the user's specific permission to the iTunes U site and resources within the site. After you have established a successful login, you can begin customizing the site, creating Course pages, and assigning user access privileges.

The diagram below illustrates the integration interaction between your web-based infrastructure and iTunes U:



Note: You can use your existing Learning Management Systems (LMS) as an authentication and authorization gateway to iTunes U, providing a familiar interface for your users, as well as accomplishing your authentication and authorization step. You should expect to perform some level of development and system integration work to get these systems working properly with iTunes U. iTunes U is not currently integrated directly with Blackboard, WebCT, Sakai, Desire2Learn, or other LMS systems, but there may be solutions available in the future. For more details, check the iTunes U support site at http://www.apple.com/support/itunes_u/.

Transferring Authorization Data to iTunes U

To successfully access iTunes U pages when a user clicks an iTunes U link in your webpage, you must transfer a cryptographically signed token data string containing user credentials and, optionally, user identity information over SSL to iTunes U.

To transfer authorization data to iTunes U:

- 1 Generate the token data string.
 - 2 Create the digital signature.
 - 3 Send the token data string and signature to iTunes U.
-

Generating the Token Data String

To establish a session when the user clicks an iTunes U link in your webpage, you must send a cryptographically signed token data string to iTunes U authorizing the user to iTunes U. This authorization token data string contains the user's credential information, the token creation time, and, optionally, information identifying the user. By default, the token data string is only valid for 90 seconds from the time you create it. Therefore, be sure to send the token, along with the signature, to iTunes U within 90 seconds of the time you create it.

Note: A user's privileges to access specific iTunes U pages or tabs depend on the credentials sent in the authorization token data string as well as the access settings you specify within iTunes U. For more information, see Chapter 4, "Controlling User Access."

To construct the authorization token data string, create a string with the concatenation of URL-encoded key-value pairs (*credentials*, *identity*, *time*) with the form: `key1=value1&key2=value2&key3=value3`.

Specify the key-value pairs in alphabetical order, such that, before URL-encoding, the token data string is:

```
credentials=foo&identity=<jdoe@example.edu>"jdoe"&time=1139331600000
```

Next, URL-encode the string. The token data string becomes:

```
credentials=foo&identity=%3Cjdoe@example.edu%3E%22jdoe%22&time=1139331600
```

Important: You must specify the key-value pairs in alphabetical order: *credentials*, *identity*, *time*.

Token Data

String Key	Token Data String Value
------------	-------------------------

credentials	
--------------------	--

A text string containing the user's attributes and unique course identifier. For example: `Learner@urn:mace:example.edu:psych101.3.200609`

You can specify up to 100 credentials per user, separated by semicolons (;). However, the text string itself cannot contain semicolons (;), backslashes (\), or control characters. If you do not want to specify any credentials, define the key as `credentials=` with no space between the empty credential and the ampersand (&) delimiter separating the next key-value pair. If you do not specify any credentials, the user can only access publicly available content.

For an example of how to implement credentials, see the credentials example later in this document written in Java.

Note: You use this exact credential information when you set up user access and privileges to pages and tabs within your iTunes U site. For this reason, you may find it helpful to create a consistent naming scheme for all your user credential information.

For information on MACE and middleware infrastructure, see <http://middleware.internet2.edu/MACE/>.

Token Data String Key	Token Data String Value
identity	An optional text string representing a specific user. For example: "Jane Doe" <janedoe@example.edu> (jdoe) [42]. If you do not want to specify any identity information, define the key as <code>identity=</code> with no space between the empty identity and the ampersand (&) delimiter separating the next key-value pair. The identity token data string key value is defined by: <pre>"displayName"<emailAddress>(username) [userIdentifier] , where:</pre> • <code>displayName</code> is the full name of the user, used to identify the user within iTunes U. For example: Jane Doe. To use special characters in the <code>displayName</code> key value, escape the character with a backslash (\). For example, to use a double quote ("), type \". • <code>emailAddress</code> is the user's email address. For example: janedoe@example.edu. The <code>emailAddress</code> key value cannot contain these characters: <, > (less than or greater than). • <code>username</code> is the user's unique login name. For example: jdoe. You can include a domain name if your institution shares information with other domains. For example: jdoe@example.edu. The <code>username</code> key value cannot contain these characters: (,) (open or closed parentheses). • <code>userIdentifier</code> is a computer-readable identifier associated with the specified user. For example: 42. The <code>userIdentifier</code> key value cannot contain these characters: [,] (open or closed brackets). You are not required to send iTunes U any user information that would violate your privacy policies. You may send an opaque or other identifier that does not contain private user information. If you send iTunes U identity token data string key values, iTunes U can provide more detailed information about your users and the actions they perform within iTunes. For example, if you send identity information to iTunes U, you can see information about the user who uploaded a track, instead of simply the time when a track was uploaded. For an example of how to implement identities, see the identity key example later in this document written in Java.
time	The time, in seconds, when you generate the token data string. Specify <code>time</code> as the number of seconds from the epoch (midnight January 1, 1970), normalized to Greenwich Mean Time (GMT). Specify <code>time</code> as base 10, without leading zeros or trailing decimals. For example: 1139331600. To ensure that your time is synchronized with the iTunes U server time, set your institution's server time to network time (NTP). iTunes U uses the token generation time to establish a login session. A session is the duration of time, 90 seconds by default, that your authentication and authorization privileges are valid. If you access iTunes U after the token generation time has expired, iTunes U directs you to the site's login page. For a site available to the public, iTunes U ignores the token data string, logs you in as a public visitor, and displays only public information in the site.

Credentials Example

```
/**
 * Combine user credentials into an appropriately formatted string.
 *
 * @param credentials An array of credential strings. Credential
 *                    strings may contain any character but ';'
 *                    (semicolon), '\\' (backslash), and control
 *                    characters (with ASCII codes 0-31 and 127).
 *
 * @return <CODE>null</CODE> if and only if any of the credential strings
 *         are invalid.
 */
public String getCredentialsString(String[] credentials) {

    // Create a buffer with which to generate the credentials string.
    StringBuffer buffer = new StringBuffer();

    // Verify and add each credential to the buffer.
    if (credentials != null) {
        for (int i = 0; i < credentials.length; ++i) {
            if (i > 0) buffer.append(';');
            for (int j = 0, n = credentials[i].length(); j < n; ++j) {
                char c = credentials[i].charAt(j);
                if (c != ';' && c != '\\' && c >= ' ' && c != 127) {
                    buffer.append(c);
                } else {
                    return null;
                }
            }
        }
    }

    // Return the credentials string.
    return buffer.toString();
}
```

Identity Key Example

```
/**
 * Combine user identity information into an appropriately formatted
 * string.
 *
 * @param displayName The user's name (optional).
 * @param emailAddress The user's email address (optional).
 * @param username The user's username (optional).
 * @param userIdentifier A unique identifier for the user (optional).
 *
 * @return A non-<CODE>null</CODE> user identity string.
 */
public String getIdentityString(String displayName, String emailAddress,
                               String username, String userIdentifier) {

    // Create a buffer with which to generate the identity string.
    StringBuffer buffer = new StringBuffer();

    // Define the values and delimiters of each of the string's elements.
    String[] values = { displayName, emailAddress,
                       username, userIdentifier };
    char[][] delimiters = { { '"', '"' }, { '<', '>' },
                           { '(', ')' }, { '[', ']' } };

    // Add each element to the buffer, escaping
```

```
// and delimiting them appropriately.
for (int i = 0; i < values.length; ++i) {
    if (values[i] != null) {
        if (buffer.length() > 0) buffer.append(' ');
        buffer.append(delimiters[i][0]);
        for (int j = 0, n = values[i].length(); j < n; ++j) {
            char c = values[i].charAt(j);
            if (c == delimiters[i][1] || c == '\\')
                buffer.append('\\');
            buffer.append(c);
        }
        buffer.append(delimiters[i][1]);
    }
}

// Return the generated string.
return buffer.toString();
}
```

Creating the Digital Signature

Since iTunes U relies on your institution's system to provide authentication and authorization information, you must use a security mechanism for iTunes U to validate the authenticity of the information it receives from you. To accomplish this, you send your information to iTunes U along with a digital signature iTunes U uses to validate the information.

You create the digital signature by using the HMAC-SHA256 algorithm to cryptographically sign the authorization token data string you generated with the secret key shared between you and Apple.

For more information on HMAC: Keyed-Hashing for Message Authentication and Secure Hash Standards, see RFC 2104 - HMAC at <http://www.faqs.org/rfcs/rfc2104.html> and http://www.governmentsecurity.org/articles/articles2/fips-180-2.pdf_fl/, respectively.

To create the digital signature:

- 1 Create a signature string combining the token data string and shared secret key.
- 2 Run the HMAC-SHA256 secure hash algorithm on the signature string (containing the token data string and shared secret key). HMAC-SHA256 takes an input of any length and generates a 32-byte output value.
- 3 Convert the HMAC-SHA256 output value into a 64-character hexadecimal representation.

For an implementation example, review the following code fragment written in Java:

```
/**
 * Generate and sign an authorization token that you can use to securely
 * communicate to iTunes U a user's identity and credentials. The token
 * includes all the data you need to communicate to iTunes U as well as
 * a creation time stamp and a digital signature for the data and time.
 *
 * @param credentials The user's credentials string, as
 *                    obtained from getCredentialsString().
 * @param identity The user's identity string, as
 *                obtained from getIdentityString().
 */
```

```

* @param time Token time stamp. The token will only be valid from
*           its time stamp time and for a short time thereafter
*           (usually 90 seconds thereafter, this "transfer
*           timeout" being configurable in the iTunes U server).
* @param key The bytes of your institution's iTunes U shared secret key.
*
* @return The authorization token. The returned token will
*         be URL-encoded and can be sent to iTunes U with
*         a <A HREF="http://www.ietf.org/rfc/rfc1866.txt">form
*         submission</A>. iTunes U will typically respond with
*         HTML that should be sent to the user's browser.
*/
public String getAuthorizationToken(String credentials, String identity,
                                   Date time, byte[] key) {

    // Create a buffer with which to generate the authorization token.
    StringBuffer buffer = new StringBuffer();

    // Generate the authorization token.
    try {

        // Start with the appropriately encoded credentials.
        buffer.append("credentials=");
        buffer.append(URLEncoder.encode(credentials, "UTF-8"));

        // Add the appropriately encoded identity information.
        buffer.append("&identity=");
        buffer.append(URLEncoder.encode(identity, "UTF-8"));

        // Add the appropriately formatted time stamp. Note that
        // the time stamp is expressed in seconds, not milliseconds.
        buffer.append("&time=");
        buffer.append(time.getTime() / 1000);

        // Generate and add the token signature.
        String data = buffer.toString();
        buffer.append("&signature=");
        buffer.append(this.hmacSHA256(data, key));

    } catch (UnsupportedEncodingException e) {

        // UTF-8 encoding support is required.
        throw new java.lang.AssertionError(
            "iTunesU.getAuthorizationToken(): "
            + "UTF-8 encoding not supported!");

    }

    // Return the signed authorization token.
    return buffer.toString();
}

/**
* Generate the HMAC-SHA256 signature of a message string, as defined in
* <A HREF="http://www.ietf.org/rfc/rfc2104.txt">RFC 2104</A>.
*
* @param message The string to sign.
* @param key The bytes of the key to sign it with.
*
* @return A hexadecimal representation of the signature.
*/
public String hmacSHA256(String message, byte[] key) {

```

```
// Start by getting an object to generate SHA-256 hashes with.
MessageDigest sha256 = null;
try {
    sha256 = MessageDigest.getInstance("SHA-256");
} catch (NoSuchAlgorithmException e) {
    throw new java.lang.AssertionError(
        this.getClass().getName()
        + ".hmacSHA256(): SHA-256 algorithm not found!");
}

// Hash the key if necessary to make it fit in a block (see RFC 2104).
if (key.length > 64) {
    sha256.update(key);
    key = sha256.digest();
    sha256.reset();
}

// Pad the key bytes to a block (see RFC 2104).
byte block[] = new byte[64];
for (int i = 0; i < key.length; ++i) block[i] = key[i];
for (int i = key.length; i < block.length; ++i) block[i] = 0;

// Calculate the inner hash, defined in RFC 2104 as
// SHA-256(KEY ^ IPAD + MESSAGE)), where IPAD is 64 bytes of 0x36.
for (int i = 0; i < 64; ++i) block[i] ^= 0x36;
sha256.update(block);
try {
    sha256.update(message.getBytes("UTF-8"));
} catch (UnsupportedEncodingException e) {
    throw new java.lang.AssertionError(
        "iTunesU.hmacSH256(): UTF-8 encoding not supported!");
}
byte[] hash = sha256.digest();
sha256.reset();

// Calculate the outer hash, defined in RFC 2104 as
// SHA-256(KEY ^ OPAD + INNER_HASH), where OPAD is 64 bytes of 0x5c.
for (int i = 0; i < 64; ++i) block[i] ^= (0x36 ^ 0x5c);
sha256.update(block);
sha256.update(hash);
hash = sha256.digest();

// The outer hash is the message signature...
// convert its bytes to hexadecimal.
char[] hexadecimals = new char[hash.length * 2];
for (int i = 0; i < hash.length; ++i) {
    for (int j = 0; j < 2; ++j) {
        int value = (hash[i] >> (4 - 4 * j)) & 0xf;
        char base = (value < 10) ? ('0') : ('a' - 10);
        hexadecimals[i * 2 + j] = (char)(base + value);
    }
}

// Return a hexadecimal string representation of the message signature.
return new String(hexadecimals);
```

Next, send both the token data string and digital signature to iTunes U.

Sending the Token Data String and Signature to iTunes U

A successful integration with iTunes U requires that you send the signed token data string over SSL to iTunes U. To do this, send an https request with corresponding parameters, along with the token data string and digital signature, to an iTunes U URL. For example:

```
https://deimos.apple.com/WebObjects/Core.woa/Browse/Example.edu.42
credentials=Administrator%40urn%3Apace%3Aitunesu.com%3Asites%3Aexample
.edu&identity=%22Jane+Doe%22+%3Cjanedoe%40example.edu%3E+%28jdoe%29+%5
B42%5D&time=1147136717&signature=597c304e90fb62067c7e3fa57fe824e77997d
d8aa96649366c5fc59104074744
```

You can use the following code example to implement the iTunes U transfer scheme:

```
/**
 * Send a request for an action to iTunes U with an authorization token.
 *
 * @param url URL defining how to communicate with iTunes U and
 *           identifying which iTunes U action to invoke and which
 *           iTunes U page or item to apply the action to. Such URLs have
 *           a format like <CODE>[PREFIX]/[ACTION]/[DESTINATION]</CODE>,
 *           where <CODE>[PREFIX]</CODE> is a value like
 *           "https://deimos.apple.com/WebObjects/Core.woa" which defines
 *           how to communicate with iTunes U, <CODE>[ACTION]</CODE> is
 *           a value like "Browse" which identifies which iTunes U action
 *           to invoke, and <CODE>[DESTINATION]</CODE> is a value like
 *           "example.edu" which identifies which iTunes U page or item to
 *           apply the action to. The destination string "example.edu"
 *           refers to the root page of the iTunes U site identified by
 *           the domain "example.edu". Destination strings for other items
 *           within that site contain the site domain followed by
 *           numbers separated by periods. For example:
 *           "example.edu.123.456.0789". You can find these strings in the
 *           items' URLs, which you can obtain from iTunes. See the
 *           iTunes U documentation for details.
 * @param token Authorization token generated by getAuthorizationToken().
 *
 * @return The iTunes U response, which may be HTML or
 *         text depending on the type of action invoked.
 */
public String invokeAction(String url, String token) {

    // Send a request to iTunes U and record the response.
    StringBuffer response = null;
    try {

        // Verify that the communication will be over SSL.
        if (!url.startsWith("https")) {
            throw new MalformedURLException(
                "ITunesU.invokeAction(): URL \""
                + url + "\" does not use HTTPS.");
        }

        // Create a connection to the requested iTunes U URL.
        HttpURLConnection connection =
            (HttpURLConnection)new URL(url).openConnection();
        connection.setUseCaches(false);
        connection.setDoOutput(true);
        connection.setRequestMethod("POST");
        connection.setRequestProperty(
            "Content-Type",
            "application/x-www-form-urlencoded; charset=UTF-8");
        // Send the authorization token to iTunes U.
```

```
// Send the authorization token to iTunes U.
connection.connect();
OutputStream output = connection.getOutputStream();
output.write(this.getBytes(token, "UTF-8"));
output.flush();
output.close();

// Read iTunes U's response.
response = new StringBuffer();
InputStream input = connection.getInputStream();
Reader reader = new InputStreamReader(input, "UTF-8");
reader = new BufferedReader(reader);
char[] buffer = new char[16 * 1024];
for (int n = 0; n >= 0;) {
    n = reader.read(buffer, 0, buffer.length);
    if (n > 0) response.append(buffer, 0, n);
}

// Clean up.
input.close();
connection.disconnect();

} catch (UnsupportedEncodingException e) {

    // iTunes U requires UTF-8 and ASCII encoding support.
    throw new java.lang.AssertionError(
        "iTunesU.invokeAction(): UTF-8 encoding not supported!");

} catch (IOException e) {

    // Report communication problems.
    throw new java.lang.AssertionError(
        "iTunesU.getResponse(): I/O Exception " + e);

}

// Return the response received from iTunes U.
return response.toString();

}
```

When iTunes U receives the https request, it validates the signed token data string using the shared secret key and the HMAC-SHA256 algorithm. If the signature is valid, iTunes U returns a response to your server. Your institution's server then forwards the response to the user's web browser, which in turn opens iTunes to the page specified, *Example.edu.42*, in the https request.

Linking to iTunes U

Each page, each tab within each page, and each track within each tab in your iTunes U site has a unique URL, making it easy for you to create links or bookmarks from your webpages or Learning Management Systems (LMS) to the requested iTunes U page.

To obtain the link to a specific iTunes U page or tab:

- 1 Open a plain text file.
 - 2 Navigate to the specific iTunes U page or tab you want.
-

- 3 Drag the navigation bar to your open text file.



Note: You can drag any segment from the navigation bar or any link in iTunes U to your desktop to create a quicklink to that page, tab, or link.

- 4 Select the URL in the text file.
- 5 Copy the selected URL and paste it into the link you create on your webpage. This is the URL you use to generate a link from your webpage to the selected page or tab within iTunes U.

Note: To obtain the link to a specific track, Control-click the track, choose Copy iTunes Music Store URL from the shortcut menu, and paste the URL into a plain text file.

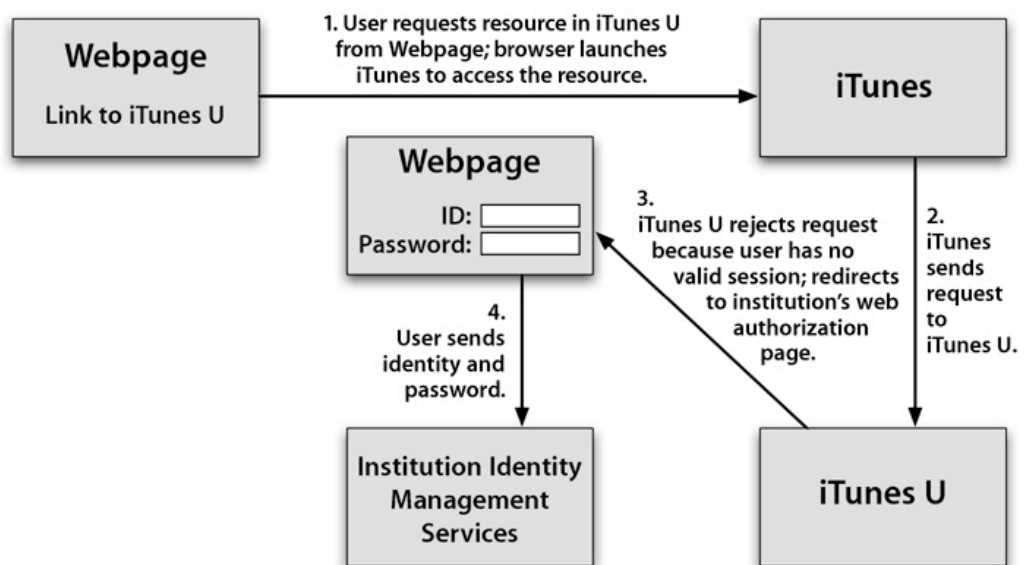
When a user clicks a link to return to a page, tab, or track, if iTunes U determines that the user is not currently logged in to an iTunes U session, the following steps occur:

- 1 iTunes U redirects the request, based on your specifications, to your institution's web authentication and authorization server, along with the URL destination information.
- 2 Your institution's web server re-authenticates the user and sends the necessary authorization token data string to iTunes U, along with the URL destination information.

Note: It is important to resend the URL destination information when you send the authorization token data string to iTunes U. Without the URL destination, iTunes U cannot display the requested page, tab, or track and, instead, displays the Welcome page for your site.

- 3 iTunes U displays the URL.

The following diagram illustrates what happens when a user clicks a link referencing iTunes U content:



Note: After step 4 in the diagram, your server validates the user and transfers authorization data to iTunes U. For more information on the next steps, see step 3 in the previous iTunes U integration interaction diagram and “Transferring Authorization Data to iTunes U” earlier in this document.

If, when you create your own iTunes U pages, you use a variable in the Course page information, you can dynamically create a well-defined URL to link to the specific Course page in iTunes U. For information on how to create a Course page, see “Creating and Editing Course Pages” in Chapter 3.

Testing Your iTunes U Integration Code

Before you send a cryptographically signed token data string to iTunes U, you should test that your code constructs the string properly and that your resulting signature matches what iTunes U expects.

For your testing purposes, Apple provides sample data you can use to test that your code produces the expected result. Using the sample data, your code should produce an identical signature to the one below. If your code is not producing the same result, check for inconsistencies using the details provided earlier in this chapter.

Using the following sample data:

- Display name=Jane Doe
- Email address=janedoe@example.edu
- Username=jdoe
- User identifier=42
- Credential=Administrator@urn:mace:itunesu.com:sites:example.edu
- Shared secret key=STRINGOFTHIRTYTWOLETTERSORDIGITS

Your resulting token data string should be the following:

```
credentials=Administrator%40urn%3Amace%3Aitunesu.com%3Asites%3Aexample.edu&identity=%22Jane+Doe%22+%3Cjanedoe%40example.edu%3E+%28jdoe%29+%5B42%5D&time=1147136717
```

And, after combining the token data string with the secret key, STRINGOFTHIRTYTWOLETTERSORDIGITS, your resulting signature value should be the following:

```
597c304e90fb62067c7e3fa57fe824e77997dd8aa96649366c5fc59104074744
```



Apple Computer, Inc.

© 2006 Apple, Computer, Inc. All rights reserved. Apple, the Apple logo, iPod, iTunes, and Mac are trademarks of Apple Computer, Inc., registered in the U.S. and other countries. Safari is a trademark of Apple Computer, Inc. iTunes Music Store is a service mark of Apple Computer, Inc., registered in the U.S. and other countries.
