WebCT 3.7 Campus Edition

Technical Reference Guide

Technical Communications

February 28, 2002

ABOUT THIS DOCUMENT

The Technical Reference Guide for WebCT 3.7 Campus Edition users and WebCT 3.7 Campus Edition Limited Use License holders is a how-to manual for carrying out advanced administration, integration and reporting tasks not available through the administrator interface. It is written for systems administrators and Web developers.

This document is separated into two main sections, one for each of the two license types available.

SECTION 1: CAMPUS EDITION LIMITED USE LICENSE

Campus Edition Limited Use License holders have access to the Standard API, which provides advanced WebCT administrative and reporting functions. Examples in this guide focus primarily on syntax and assume a strong Web programming background. For more detailed examples, download the Practical Examples document from the WebCT Documentation, API Guides section at download.webct.com.

An overview of Section 1 follows:

Chapter 1: User Authentication

- Describes methods for providing secure access to WebCT.

Chapter 2: Standard API

- Describes how to add, update, and delete one or multiple users.
- Describes how to find users in the global or student database.
- Describes how to change WebCT IDs.

SECTION 2: CAMPUS EDITION

Campus Edition users have access to the IMS API and the Standard API. The IMS API enables integration with existing campus systems, such as student information systems and portals. The Standard API allows access to advanced WebCT administrative and reporting functions.

Examples in this guide focus primarily on syntax and assume a strong Web programming background. For more detailed examples, download the Practical Examples document from the WebCT Documentation, API Guides section at download.webct.com.

An overview of Section 2 follows:

Chapter 1: User Authentication

- Describes methods for providing secure access to WebCT, using one or more authentication sources.

Chapter 2: Automatic Signon from Other Systems

- Describes how to implement automatic signon.

Chapter 3: Overview of the Application Programming Interfaces

- Describes the functional and operational differences between the IMS Enterprise API and the Standard API.

Chapter 4: IMS Enterprise API

- Describes how to import data into the WebCT database, export data from the WebCT database, and how to set the IMS ID for group and person objects.

Chapter 5: Standard API

- Describes how to add, update, and delete one or multiple users.
- Describes how to find WUUIs and how to find users in the global or student database.
- Describes how to Change WebCT IDs.
- Describes how to export myWebCT in XML format.

SECTION 1: CAMPUS EDITION LIMITED USE LICENSE

CHAPTER 1 USER AUTHENTICATION

WebCT 3.7 Campus Edition (CE) provides two major methods for user authentication:

Browser Based Authentication	• Users are authenticated through a browser dialog box that prompts for a username and password. The username and password are verified against WebCT's internal database. If the user is authorized, a Basic Authentication Header is provided. Subsequent page accesses to WebCT are authorized according to the browser header. • This authentication method is used in previous versions of WebCT.
Ticket Based Authentication	• Users are authenticated through a logon page that prompts for a username and password. The username and password are verified against WebCT's internal database. If the authentication is successful, the user is issued a browser cookie that serves as a ticket. Subsequent page accesses to WebCT are authorized according to the ticket.

CHOOSING AN AUTHENTICATION METHOD

Certain features of WebCT 3.7 CE require ticket-based authentication, including:

- Logout
- Server lockdown
- Session timeout
- Customizable logon page

Browser based authentication is primarily provided in WebCT 3.7 as a legacy option. Choose this method of authentication if:

- Your institution has an information technology policy forbidding the use of applications that employ browser cookies.
- It is critical that the user interface of WebCT remain the same as previous versions.

BROWSER BASED AUTHENTICATION PROCESS

Browser-based authentication has served as the standard authentication method for all previous versions of WebCT. When a user a requests a URL, authentication of the user occurs as follows:

1. The Web server checks to see if the requested URL requires authorization.
2. If none is required (e.g., a user requests a course Welcome Page), then the Web server delivers the page to the browser.
3. If authorization is required, the Web server checks to see if the user has already provided a username and password by checking to see if a valid Basic Authentication Header was provided in the request. If the header is valid, the page is delivered.
4. If the Basic Authentication Header is invalid, or no header is provided, the user is prompted with a username and password dialog box. The cycle is then repeated.

TICKET BASED AUTHENTICATION PROCESS

When a user requests a URL, authentication of the user occurs as follows:

5. The Web server checks to see if the requested URL requires authorization.
6. If none is required, the Web server delivers the page to the browser.
7. If authorization is required, WebCT checks for a valid ticket.
8. If a valid ticket is found (i.e. the user has been authenticated and is authorized for the resource), the page is delivered to the browser.
9. If a ticket is not found, a logon form is delivered to the browser. The user submits the form and WebCT authenticates the user, issuing their browser a cookie. The URL is re-requested and the cycle repeats.

HOW WEBCT GENERATES TICKETS

WebCT tickets (in the form of browser cookies) contain the following information:

- Username
- Encrypted Password (DES encryption)
- Timestamp (UNIX Epoch format)
- Message Authentication Code (MAC)

The MAC is generated in three steps:

1. The username, encrypted password, timestamp, user agent information (if sent), and a shared secret value are concatenated.
2. The concatenated string is encrypted with the MD5 algorithm.
3. The encrypted string is encrypted a second time with the MD5 algorithm.

IMPLEMENTING TICKET BASED AUTHENTICATION

With ticket-based authentication, you authenticate using WebCT's internal database.

1. From the *Admin* toolbar, click **Server Mgmt**. The *Server Mgmt* toolbar appears.
2. From the *Server Mgmt* toolbar, click **Settings**. The *Administrator Settings* screen appears.
3. Under *User Authentication*, select *Use ticket based authentication*.
4. Choose whether the **Logout** link should appear in the course *Menu Bar*:
   - To display the **Logout** link, select *Display Logout link in course Menu Bar*.
   - To hide the **Logout** link, deselect *Display Logout link in course Menu Bar*. **Note:** If you run WebCT in a framed environment (such as a portal) where a logout link or "Return to Portal" link already exists, you may prefer to hide the **Logout** link.

5. In the *Ticket shared secret value* text box, either leave the shared secret value that was automatically generated by WebCT or enter a new shared secret value. For security reasons, the default value "secret" does not work. The secret value

    - is case-sensitive
    - cannot exceed 256 characters
    - cannot contain tab or other control characters
    - should not contain end-of-line characters. **Note:** By default, the UNIX text editors vi and pico automatically add end-of-line characters. Check the file size to ensure that the number of characters equals the number of bytes.

6. In the *Tickets remain valid for* text box, enter the number of minutes until ticket time-out. This value controls the expiry time of the ticket based on the user's last access and therefore affects how long a user can stay logged on while inactive. The default is 180 minutes.
7. Choose whether to allow WebCT authentication across a domain. Authentication across a domain allows users to access all servers in the domain, without having to re-authenticate for each one.

    - To allow authentication across a domain:
        a) Select *Allow WebCT authentication across a domain*.
        b) In the *Please specify your domain* text box, enter the domain name. The domain name must have a period in front of it. Example: `.webct.com`
    - To disallow authentication across a domain, select *Do not allow WebCT authentication across a domain*.

8. Under *User is authenticated using*, from the drop-down list for the authentication source that you are using, select *First*. **Note:** With a Limited Use License, only the WebCT internal database can be used as the authentication source.
9. For all other authentication sources, select *Never*.
10. Scroll to the bottom of the screen and click **Update**.

CHAPTER 2
STANDARD API

The Standard API gives administrators and developers access to the WebCT databases via command line or Web-based interfaces, to perform a variety of administration and reporting tasks.

OVERVIEW OF THE STANDARD APPLICATION PROGRAMMING INTERFACE

Application Programming Interfaces (APIs) allow users and other systems to directly interface with WebCT without the graphical user interface. This chapter describes the WebCT 3.7 CE (proprietary) Standard API. This API has two interfaces, a command line interface and a Web-based interface.

CHOOSING THE APPROPRIATE INTERFACE FOR YOUR REQUIREMENTS

WebCT provides two interfaces to of the Standard API: a command line interface, and a Web-based interface. Choosing an interface is not a one-time decision; it will vary depending on the task that you need to accomplish.

Use the following table as a guide for choosing the best interface for your task.

Task	Suggested Interface
Processing multiple records simultaneously	command line
Processing a single record	command line
Integrating systems that are on the same physical server and run as the same user as WebCT	command line
Debugging	command line
Integrating external system with WebCT (e.g., you want to integrate your institution's SIS with WebCT)	Web-based

FUNCTIONALITY IN THE STANDARD API

The Standard API allows you to manipulate two separate databases within WebCT, the global database and the student database.

The global database contains the central listing of all users on the WebCT server. By default, the global database contains the WebCT ID, Password, First Name, Last Name, Courses, and Registered Courses fields. All users must have an entry in this database in order to access a course.

The student database is a term for a collection of databases specific to a course. Every WebCT course has its own student database that contains, by default, the User ID, Password, First Name, and Last Name fields. The information in the student database can be viewed most readily by looking at the designer interface of Manage Students.

Generally, a WebCT ID is linked to a User ID for each course that a user is enrolled in. Users can have different User IDs from their WebCT IDs, as well as different First Name and Last Name data in the student and global databases.

The functionality of the Standard API can be divided into the following basic categories:

Adding Users	• Adding a single user to the global database or student database • Adding multiple users to the global database or student database
Updating Users	• Updating a single user in the global database or student database • Updating multiple users in the global database or student database • Updating user types
Deleting Users	• Deleting a single user from the global database or student database • Deleting multiple users from the global database or student database
Finding Users	• Finding a user in the global database or student database
Changing WebCT IDs	• Changing a single user's WebCT ID • Changing multiple users' WebCT IDs

IMPLEMENTING THE STANDARD API

COMMAND LINE INTERFACE (WEBCTDB)

The command line interface to the standard API provides a simple interface to the WebCT API. The executable file `webctdb`, is located in the directory `<install_dir>/webct/webct/generic/api/.`

SYNTAX

The general syntax for each of the Standard API operations is as follows:

Operation	Field Names
add	<db> <course> <fieldsData_pair_list> <separator> [encrypted]
delete	<db> <course> <WebCT_ID | user_id>
changeid	<db> <course> <fieldsData_pair_list> <separator>
update	<db> <course> <fieldsData_pair_list> <separator> [encrypted]
find	<db> <course> <WebCT_ID | user_id> <separator> [user_type]

Operation	Field Names
fileadd	<db> <course> <filename> <separator> [encrypted]
fileupdate	<db> <course> <filename> <separator> [encrypted]
filedelete	<db> <course> <filename>
filechangeid	<db> <course> <filename> <separator>

Details about each field name are provided below:

Field Name: `db`
Value: `global or student`
Example: `global`
Description: This is the name of the database, either global database or student database.

Field Name: `course`
Value: `Course ID`
Example: `cs100`
Description:
- Required for student database operations.
- For global database operations, enter the placeholder value xxxx.

Field Name: `fieldsData_pair_list`
Value: A double quote-enclosed list of field-data pairs in the form: field_name=data_value.
Example: `"WebCT ID=student1"`
Description:
- The field names must exist in the WebCT global database or student database. The *separator* must be inserted between each of the field-data pairs.
- For the global database, the optional fields Courses and Registered Courses are available for adding and/or modifying courses and registered courses to which a global user belongs. The values for these fields can be a colon-separated list of course IDs for Courses or course names for Registered Courses. For example, `Courses=cs100:psyc100:math100.` If you also specify a user type with the course, this is separated from the course ID by a semicolon, for example, `Courses=cs100;D:psyc100;TA`. **Note**: The default user type is (S)tudent.
- A user can be added as a primary designer or as a secondary designer. The first WebCT ID added to the course as a designer becomes the primary designer; every subsequent designer becomes a secondary designer.

Field Name: `fieldsData_pair_list (cont.)`

Description: **Note**: The following are reserved words in the fieldsData_pair_list:

- Login ID (this is old terminology, and is supported for backward compatibility only. It has the same meaning as User ID).
- User ID (the User ID of a student in a course)
- Password (the password of the global user or the student)
- Global ID (this is old terminology, and is supported for backward compatibility only. It has the same meaning as WebCT ID).
- WebCT ID (WebCT ID of a global user)
- First Name (first name of the global user or the student. It is one of the reserved columns in both the global and student databases)
- Last Name (last name of the global user or the student. It is one of the reserved columns in both the global and student databases)
- Courses (the list of WebCT courses for a global user). If you populate this field through the API, the course must already exist on the WebCT server.
- Registered Courses (the list of courses maintained by the registrar for a global user. These courses may or may not have a WebCT course.)
- Thumbprint (internal data and cannot be modified)
- LockPID (internal data and cannot be modified)
- #User Type (internal data. This can be modified through the API)
- #Email (internal data and cannot be modified)
- #Password Question (internal data and cannot be modified)
- #Password Answer (internal data and cannot be modified)

Note: Reserved words are case sensitive.

Field Name: `separator`

Value: Any alphanumeric string representing the separator between data pairs in the fieldsData_pair_list.

Example: "," (comma)

Description: Delimiter used to separate data items. You must declare what value you will be using as a delimiter for the operations add, delete, changeid, update, and find.

Note: For the global database, the colon and semi-colon are not allowed as separators.

Field Name: `user_type`

Value: `user_type`

Example: `user_type`

Description: Only used with the `find` operation on the global database; return value of user_type is one of three users types, S,D,TA (for Student, Designer, Teaching Assistant)

Field Name:	`encrypted`
Value:	`encrypted`
Example:	`encrypted`
Description:	- Only used with the add, update, fileadd and fileupdate operations. - The password must be encrypted using the standard UNIX DES encryption method or the newly added or modified users may not be able to access WebCT. - Add to the end of the command line to indicate that the passwords are passing in encrypted form.

FUNCTIONS

ADDING USERS

Users can be added to the global database or student databases. However, in general, you should add users to the global database and use the Courses field to add them to each course. This method is simpler and automatically links the WebCT ID to each User ID.

ADDING A SINGLE USER TO THE GLOBAL DATABASE

Operation = `add`
• The fieldsData_pair_list must include both the WebCT ID and Password fields. • You can specify the user type (S for student, D for designer, TA for teaching assistant). If you don't specify a user type, the user type defaults to (S)tudent. If the user type is specified as (D)esigner and there is no existing designer, the user is added as the primary designer. If there is an existing designer, the user is added as secondary designer.

Syntax

add
<db> <course> <fieldsData_pair_list> <separator> [encrypted]

Example

Add a user named Justin Case to the global database as a designer for cs100; a teaching assistant for cs200; and as a student in cs810:

Enter the command:

```
webctdb add global xxxx "WebCT ID=jcase,Password=1234,
      First Name=Justin,Last Name=Case,Courses=cs100;D:cs200;TA:cs810;S" ","
```

ADDING A SINGLE USER TO THE STUDENT DATABASE

Operation = `add`
• The fieldsData_pair_list must include both the User ID and Password fields.

Syntax

add
<db> <course> <fieldsData_pair_list> <separator> [encrypted]

Example

Add a user named Bailey Wick to the student database for course cs100:

Enter the command:

```
webctdb add student cs100 "User ID=bwick,Password=1234,
      First Name=Bailey,Last Name=Wick" ","
```

ADDING MULTIPLE USERS TO THE GLOBAL DATABASE

Operation = fileadd

- filename is either a full absolute path or a relative path from the current directory to the file. For example, if the file is located in the current directory, then filename is only the name of the file. A file extension, such as `.txt`, is recommended.
- The file must be in plain text. The first line of the file must be the field names separated by the separator value. The rest of the file contains the data, one record per line. Data should be in the same order as the field names, separated by the separator value. There must be no spaces between the data and the separators.
- If the user exists in the database, `fileadd` will send an error message to STDOUT. The user record will not be changed; the process will skip to the next record in the file.
- An optional `encrypted` argument can be added at the end of the command line to indicate that the passwords are passing in an encrypted form. The passwords should be encrypted using the standard UNIX DES encryption method or the newly added or modified users may not be able to access WebCT.

Syntax

fileadd
<db> <course> <filename> <separator> [encrypted]

Example

Add users to the global database from a text file named `users.txt`.

SAMPLE USERS.TXT FILE:

```
WebCT ID,Password,Last Name,First Name
jsmith,9876,Smith,John
jbrown,2345,Brown,Jane
bfawlty,8765,Fawlty,Basil
arigsby,5432,Rigsby,Arthur
```

Enter the command:

```
webctdb fileadd global xxxx users.txt ",”
```

ADDING MULTIPLE USERS TO THE STUDENT DATABASE

Operation = fileadd

- filename is either a full absolute path or a relative path from the current directory to the file. For example, if the file is located in the current directory, then filename is only the name of the file. A file extension, such as `.txt`, is recommended.
- The file must be in plain text. The first line of the file must be the field names separated by the separator string. The rest of the file contains the data, one record per line. Data should be in the same order as the field names, separated by the separator value. There must be no spaces between the data and the separators.
- If the user exists in the database, `fileadd` will send an error message to STDOUT. The user record will not be changed in the database; the process will skip to the next record in the file.
- An optional `encrypted` argument can be added at the end of the command line to indicate that the passwords are passing in an encrypted form. The passwords should be encrypted using the standard UNIX DES encryption method or the newly added or modified users may not be able to access WebCT.

Syntax

fileadd
<db> <course> <filename> <separator> [encrypted]

Example

Add students whose records are stored in the file `class.txt` to the course cs100.

SAMPLE CLASS.TXT FILE

```
User ID,Password,Last Name,First Name,
jsmith,9876,Smith,John
jbrown,2345,Brown,Jane
bfawlty,8765,Fawlty,Basil
arigsby,5432,Rigsby,Arthur
```

Enter the command:

```
webctdb fileadd student cs100 class.txt ","
```

UPDATING USERS

UPDATING A SINGLE USER IN THE GLOBAL DATABASE

Operation = update

- The fieldsData_pair_list must include the WebCT ID.
- Empty fields are not changed.
- If the field value is "_DELETE_", the value will be set to null
- The Standard API behavior when updating the Courses and Registered Courses field is to always overwrite the field. If you supply a Courses field in your update, the user's WebCT ID will be linked to the courses that you supply, and unlinked from any pre-existing courses that you do not supply.
- You can update a user type by specifying a different one.
- `update` cannot be used to modify quiz scores.
- An optional `encrypted` argument can be added at the end of the command line to indicate that the passwords are passing in an encrypted form. The passwords should be encrypted using the standard UNIX DES encryption method or updated users may not be able to access WebCT.

Syntax

update
<db> <course> <fieldsData_pair_list> <separator> [encrypted]

Example

For the student Justin Case, password 1234, with the following courses: cs100(D) cs200(TA) cs810(S), update the password in the global database and update the courses so that only cs100 remains.

Enter the command:

```
webctdb update global xxxx "WebCT ID=jcase,Password=abcd,
      Courses=cs100" ","
```

UPDATING A SINGLE USER IN THE STUDENT DATABASE

Operation = update

- The fieldsData_pair_list must include the User ID field.
- Empty fields are not changed.
- If the field value is "_DELETE_", the value will be set to null
- The Standard API behavior when updating the Courses and Registered Courses field is to always overwrite the field. If you supply a Courses field in your update, the user's WebCT ID will be linked to the courses that you supply, and unlinked from any pre-existing courses that you do not supply
- You can update a user type by specifying a different one.
- `update` cannot be used to modify quiz scores.
- An optional `encrypted` argument can be added at the end of the command line to indicate that the passwords are passing in an encrypted form. The passwords should be encrypted using the standard UNIX DES encryption method or the updated users may not be able to access WebCT.

Syntax

update

<db> <course> <fieldsData_pair_list> <separator> [encrypted]

Example

To update the student Bailey Wick, first name, last name, and password of password "1234".

Enter the command:

```
webctdb update student cs100 "User ID=bwick,Password=abcd,
      First Name=Bailie,Last Name=Wicke" ","
```

UPDATING MULTIPLE USERS IN THE GLOBAL DATABASE

Operation = fileupdate

- filename is either a full absolute path or a relative path from the current directory to the file. For example, if the file is located in the current directory, then filename is the name of the file. A file extension, such as `.txt`, is recommended.
- The file must be in plain text. The first line of the file must be the field names separated by the separator. The rest of the file contains the data, one record per line. Data should be in the same order as the field names, separated by the separator value. There must be no spaces between the data and the separators.
- An optional `encrypted` argument can be added at the end of the command line to indicate that the passwords are passing in an encrypted form. The passwords should be encrypted using the standard UNIX DES encryption method or the newly added or modified users may not be able to access WebCT.
- `fileupdate` will add a user if they do not exist in the database.
- Empty fields will not be changed.
- For `fileupdate`, if the value field value is "_DELETE_", the value will be set to null
- The Standard API behavior when updating the Courses and Registered Courses field is to always overwrite the field. If you supply a Courses field in your update, the user's WebCT ID will be linked to the courses that you supply, and unlinked from any pre-existing courses that you do not supply

Syntax

fileupdate

<db> <course> <filename> <separator> [encrypted]

Example

Change the names of a group of users whose updates are contained in the file `updates.txt`.

SAMPLE UPDATES.TXT FILE:

```
WebCT ID,Last Name,First Name
jsmith,Smith,Jerry
jbrown,Brown,Janet
bfawlty,Fawlty,Brian
arigsby,Rigsby,Alan
```

Enter the command:

```
webctdb fileupdate global xxxx updates.txt ","
```

UPDATING MULTIPLE USERS IN THE STUDENT DATABASE

Operation = fileupdate

- filename is either a full absolute path or a relative path from the current directory to the file. For example, if the file is located in the current directory, then filename is the name of the file. A file extension, such as `.txt`, is recommended.
- The file must be in plain text. The first line of the file must be the field names separated by the separator. The rest of the file contains the data, one record per line. Data should be in the same order as the field names, separated by the value of the separator. There must be no spaces between the data and the separators.
- An optional `encrypted` argument can be added at the end of the command line to indicate that the passwords are passing in an encrypted form. The passwords should be encrypted using the standard UNIX DES encryption method or the newly added or modified users may not be able to access WebCT.
- `fileupdate` will add a student or user if they do not already exist in the database.
- Empty fields will not be changed.
- For `fileupdate`, if the field value is "_DELETE_", the value will be set to null
- `fileupdate` overwrites the data fields being changed; it does not append.

Syntax

fileupdate
<db> <course> <filename> <separator> [encrypted]

Example

Change the names of students in the course cs100 using updates contained in the file `updates.txt`.

SAMPLE UPDATES.TXT FILE:

```
User ID,Last Name,First Name
jsmith,Smith,Jerry
jbrown,Brown,Janet
bfawlty,Fawlty,Brian
arigsby,Rigsby,Alan
```

Enter the command:

```
webctdb fileupdate student cs100 updates.txt ","
```

DELETING USERS

DELETING A SINGLE USER FROM THE GLOBAL DATABASE

Operation = delete
• global_id is the ID of the user to be deleted from the global database. **Note**: Depending on the *User Data* setting in the administrator interface, the student's data may also be deleted from the student database.

Syntax

delete
<db> <course> <WebCT_ID | user_id>

Example

Delete the global database record for the user whose WebCT ID is jcase. **Note**: The student will be denied access to all the courses listed in their global database record. Depending on the *User Data* setting in the administrator interface, the student's data may also be deleted from the student database.

Enter the command:

```
webctdb delete global xxxx jcase ","
```

DELETING A SINGLE USER FROM THE STUDENT DATABASE

Operation = delete
• user_id is the ID of the student to be deleted from the student database.

Example

Delete the record for the student in the cs100 course whose User ID is bwick.

Enter the command:

```
webctdb delete student cs100 bwick ","
```

DELETING MULTIPLE USERS FROM THE GLOBAL DATABASE

Operation = filedelete
• filename is the either a full absolute path or a relative path from the current directory to the file. For example, if the file is located in the current directory, then filename is simply the name of the file. A file extension, such as `.txt`, is recommended. • If the user does not exist in the database, `filedelete` will send an error message to STDOUT. The process will skip to the next record in the file.

Syntax

filedelete
<db> <course> <filename>

Example

Delete users from the global database using a text file `deleteusers.txt`.

SAMPLE DELETEUSERS.TXT FILE:

```
jsmith
jbrown
bfawlty
arigsby
```

Enter the command:

```
webctdb filedelete global xxxx deleteusers.txt ","
```

DELETING MULTIPLE USERS FROM THE STUDENT DATABASE

Operation = filedelete

- filename is the either a full absolute path or a relative path from the current directory to the file. For example, if the file is located in the current directory, then filename is simply the name of the file. A file extension, such as `.txt`, is recommended.
- If the user does not exist in the database, `filedelete` will send an error message to STDOUT The process will skip to the next record in the file.

Syntax

filedelete
<db> <course> <filename>

Example

Delete students whose records are stored in the file `delete.txt` from the course cs100.

SAMPLE DELETE.TXT FILE:

```
jsmith
jbrown
bfawlty
arigsby
```

Enter the command:

```
webctdb filedelete student cs100 delete.txt
```

FINDING USERS

FINDING A USER IN THE GLOBAL DATABASE

Operation = find
• WebCT ID is the WebCT ID of the user in the global database. • Separator is the separator of the output data, which is sent to STDOUT in the same format as the fieldsData_pair_list. • If the field name user_type is specified in a global database query, the user type (S,D,TA) will be included in the result.

Syntax

find
<db> <course> <WebCT_ID | user_id> <separator> [user_type]

Example

Find the global database record, including user type, for the user with the WebCT ID jcase.

Enter the command:

```
webctdb find global xxxx jcase "," user_type
```

If the command is successfully executed:

```
Success: WebCT ID=jcase,First Name=Justin,
      Last Name=Case,Courses=cs100;D:cs200;TA:cs810;S
```

FINDING A USER IN THE STUDENT DATABASE

Operation = find
• user_id is the User ID of the student in the student database. • Separator is the separator of the output data, which is sent to STDOUT in the same format as the fieldsData_pair_list.

Syntax

find
<db> <course> <WebCT_ID | user_id> <separator> [user_type]

Example

Find the student in the cs100 course whose User ID is bwick.

Enter the command:

```
webctdb find student cs100 bwick ","
```

If the command is successfully executed:

```
Success:First Name=Bailie,Last Name=Wicke,User ID=bwick
```

CHANGING WEBCT IDS

CHANGING A SINGLE USER'S WEBCT ID

Operation = changeid
• `changeid` can only be used on the global database. • old_id is the WebCT ID to be changed. • new_id is the new WebCT ID.

Syntax

changeid
<db> <course> <fieldsData_pair_list> <separator>

Example

Change Justin Case's WebCT ID from jcase to jicase.

Enter the command:

```
webctdb changeid global xxxx "Old ID=jcase,New ID=jicase" ","
```

CHANGING MULTIPLE USERS' WEBCT IDS

Operation = filechangeid
• filename is either a full absolute path or a relative path from the current directory to the file. For example, if the file is located in the current directory, then filename is simply the name of the file. A file extension, such as `.txt`, is recommended. • The first line of the data file should be the field names Old ID and New ID, separated by the separator_file string. The rest of the file contains the data, one record per line. Data should be in the same order as the field names, separated by the separator value. **Note**: The field name Old ID does not exist in the databases. • If the user does not already exist in the database, `filechangeid` will send an error message to STDOUT The process will skip to the next record in the file.

Syntax

filechangeid
<db> <course> <filename> <separator>

Example

Change the WebCT IDs of a group of users contained in a file `changeusers.txt`.

SAMPLE CHANGEUSERS.TXT FILE:

```
Old ID,New ID
jsmith,jtsmith
jbrown,jkbrown
```

```
bfawlty,befawlty
arigsby,aurigsby
```

Enter the command:

```
webctdb filechangeid global xxxx changeusers.txt ","
```

WEB-BASED INTERFACE (SERVE_WEBCTDB)

The Web-based Standard API allows data in the WebCT global database and student databases to be queried and manipulated by remote servers. For example, the Web-based interface could be used to make changes to global database records based on registration changes driven by events on another system. It could also be used to create a custom administrator's interface.

Implementing the Web-based interface involves two steps.

1. Setting the API shared secret value
2. Developing a program to generate an HTTP request

Step 1 can be carried out by a WebCT administrator who has basic knowledge of the WebCT file system. Step 2 requires an experienced Web developer.

1. SETTING THE API SHARED SECRET VALUE

The shared secret value is a key component of allowing external servers to automatically sign on users to WebCT. The shared secret value is used to create a Message Authentication Code (MAC) from the submitted data. When WebCT receives a request, it decodes the shared secret value from MAC using the submitted data. If the decoded shared secret value is the same as the one stored locally, the request is considered authentic and is processed. You can set the shared secret value by performing the following steps:

1. Using a text editor, open the file `<webct_install_directory>/webct/webct/generic/api/api_secret`
2. Change the first line of the file to your desired secret. (For security reasons, the default value "SECRET" does not work). You should note the following about the shared secret value.
   - It cannot exceed 256 characters.
   - It cannot contain tab, or other control characters.
   - It should not contain end-of-line characters. **Note**: By default, the UNIX text editor vi and pico automatically add end-of-line characters. Check the file size to ensure that the number of characters equals the number of bytes.
   - It is case-sensitive
3. Save the file.

Because the shared secret value has such a critical role, choose it carefully.

Tips for Shared Secrets

- Make your shared secret value difficult to guess by making it lengthy and by including a combination of numbers and upper and lower case characters.
- Change your shared secret value at regular intervals.
- On remote systems, place shared secret values in secure directories.

2. Developing a Program to Generate an HTTP Request

Developing a program to generate an http request is the most substantive part of implementing the Web-based standard API. The program must:

- Generate a Message Authentication Code (MAC)
- Assemble a properly formatted http request
- Process any data being returned

Creating Message Authentication Codes

Because the Web interfaces to the Standard API reside in public directories, Message Authentication Codes (MACs) are required to ensure that only messages from trusted servers are processed.

WebCT provides three options to assist you in creating MACs:

1. A C function that you may integrate and compile into your C program
2. An executable file to which you make a system call from your program
3. Instructions for generating a MAC using a language of your choice

Option 1: Using the get_authentication C function

The get_authentication function generates a MAC from an array of data and a shared secret value.

The source code necessary to use the C function is located in `<webct_install_directory>/webct/webct/generic/api/security/`

A test program, which contains a Makefile for UNIX based systems, is also provided.

The file `api_security.c` contains the `get_authentication` function.

get_authentication	Generates a MAC from an array of data and a shared secret value
Syntax	char* get_authentication (int i, char* data[], char* secret, char* encrypted_data)
Returns	32 byte alphanumeric MAC

Parameter	Description
`i`	The number of elements in the array data[].
`data`	Array of all values to be used in generating the MAC. The data should not be URL encoded.
`secret`	The shared secret value.
`encrypted_data`	The memory location of the MAC. It must be at least 32 bytes long.

OPTION 2: USING THE MESSAGE AUTHENTICATION CODE GENERATOR (GET_MD5 EXECUTABLE)

The Message Authentication Code generator generates a MAC from a shared secret value and a string consisting of the IMS ID, a timestamp, and a destination URL.

Use the Message Authentication Code generator (an executable called get_md5) if you are not working in the C language, or do not want to create a function to create the MAC. You can make a system call to get_md5 from your program and have the authentication string returned. The get_md5 executable has no dependencies on WebCT and can be copied to other servers as required. If you need a get_md5 executable for an operating system other than the one your WebCT server is running on, you can download several pre-compiled binaries for other operating systems from `http://download.webct.com`

get_md5	Generates a MAC from a shared secret value and a string to be encrypted (consisting of the IMS ID, a timestamp, and a destination URL).
Syntax	get_md5 <shared_secret_filename> <string_to_encrypt>
Returns	32 byte alphanumeric MAC

Parameter	Description
`shared_secret_filename`	The filename where the shared secret value is stored.
`string_to_encrypt`	The string to be encrypted. The string should not be URL encoded.

An example of using the Message Authentication Code generator to generate a MAC from a shared secret value and the string described above follows.

Enter the command:

get_md5 api_secret 2A508D8EB5EB2D596DD937E2B8835100 982187291 http://webct.institution.edu:8900/SCRIPT/ENGL100-001/scripts/serve_home

OPTION 3: CREATE A MAC USING A LANGUAGE OF YOUR OWN CHOICE

If you want to create MACs within your code, (e.g. you are writing your code in Java and don't want to make a system call), you can create a MAC with the following procedure:

1. Calculate the total of the ASCII values of all the characters in the data.
2. Convert the total of the ASCII values into a string.
3. Append the shared secret value.
4. Encrypt the string into a 16-byte string using the MD5 algorithm.
5. Convert the 16-byte string into a 32-byte alphanumeric string to make it URL-friendly.

ASSEMBLING THE HTTP REQUEST

There are several options for assembling an http request to the Web-based standard API. The option you choose will be based on your programming language of choice, and how you want to communicate with the Web server. You can issue API commands in several ways, including:

- Socket programming directly with the Web server
- Using a library which simulates a user agent
- Assembling a GET request and refreshing a browser window with the query string.

In Perl, you have the option of communicating directly with the Web server using the IO::Socket module included with most basic distributions, or installing and using a module such as LWP which simulates a user agent (e.g. a Web browser). Similar modules are available for most popular languages such as C or Java.

If you wish to refresh a user's browser window with a query string, you can do so using the "Location" http header, HTML meta tags, or using JavaScript's location.replace method.

SYNTAX

The general syntax for a Web-based request to the Standard API is as follows:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=<operation>&DB=<db>
      &COURSE=<course_id | placeholder>&AUTH=<32_byte_mac>
      [&User%20ID=<user_id> | &WebCT%20ID=<webct_id>][&IMS%20ID=<ims_id>]
      [&USER%20TYPE=<1_or_0>][&ENCRYPTED=<1_or_0>][&field1=<field1>]
      [&fieldn=<fieldn>]HTTP/1.0
```

where:

Key	Value	Description
OPERATION	add	Adds a user to the global or student database. If the user already exists, an error is returned.
	update	Updates an existing user in the global or student database. If the user does not exist, this operation returns an error.

Key	Value	Description
	delete	Deletes a single user from the global or student database.
	find	Finds the user record based on the User ID (if searching the student database) or WebCT ID (if searching the global database).
	changeid	Changes a WebCT ID.
	homearea_xml	Exports a user's *myWebCT* in XML format.

Notes:

The Standard API can accept GET or POST requests. POST requests can put their key/value pairs in the query string or in the body of the message in the appropriate format (see the W3C HTML 4.01 Specification at http://www.w3.org/TR/html401/interact/forms.html#h-17.13.4)
Requests must be URL encoded (e.g. spaces should be replaced with %20)
Key/value pairs may appear in any order
Syntax examples represent http requests directly to the Web server. If you are using a programming module to create your requests (such as LWP in Perl), many details of the request may be transparent to you.

FUNCTIONS

ADDING USERS

ADDING A USER TO THE GLOBAL DATABASE

Add operations have the following syntax:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=add&DB=global
      &COURSE=<placeholder>&AUTH=<32_byte_mac>&WebCT%20ID=<webct_id>
      &Password=<password>[&field1=<field1>][&fieldn=<fieldn>]
      [&ENCRYPTED=<1_or_0>]HTTP/1.0
```

where:

Key	Value	Description
COURSE	Any alphanumeric string	This is a required placeholder value. You may use any alphanumeric value, but ensure that you use it in the calculation of the MAC.
AUTH	32 byte MAC	This is the 32 byte hexadecimal string generated using the get_authentication C code, the Message Authentication Code (MAC) generator, or using custom code.

Key	Value	Description
WebCT ID	WebCT ID	The WebCT ID of the user being added. WebCT IDs can contain alphanumeric strings, underscores, and periods.
Password	Password	The password to be used for the user being added. Passwords can consist of any alphanumeric string. The API does not enforce minimum password lengths.
Field1 … Fieldn (optional)	Data associated with the column specified as a key.	Any of the default columns within the global database (First Name, Last Name, Courses, Registered Courses) can be modified using the syntax ColumnName=Value. Administrator-created columns can also be modified using this method.
ENCRYPTED (optional)	1	Enables pre-encrypted password support. With the encrypted argument set to 1, you should pass passwords encrypted with the standard UNIX DES method when using this setting.
	0 (default)	Disables pre-encrypted password support (default). In this mode, passwords should be submitted as clear-text.

Note: The Courses field uses a colon as a delimiter between courses, and a semicolon as a delimiter between user types. Thus the string "HKIN100;D:HKIN200;TA:HKIN300;S" indicates that a user is to be added to HKIN100 as a designer, HKIN200 as a teaching assistant and HKIN300 as a student. If no user type is specified, WebCT will default to adding the user as a student. Similarly, the Registered Courses field is colon delimited. See the System Administrators Guide for more information on the Courses and Registered courses field.

Example

Add a user to the global database, and enroll them in the course ENGL100 as a designer, ENGL560 as a student, and ENGL477 as a teaching assistant.

```
GET  /webct/public/serve_webctdb?OPERATION=add&DB=global&COURSE=xxxx
      &AUTH= EB1A09F0BB299C23E99A5978587F49C1&WEBCT%20ID=pinto
      &PASSWORD=an1mal&FIRST%20NAME=Larry&LAST%20NAME=Kroger&
      COURSES=ENGL100;D:ENGL560:ENGL477;TA HTTP/1.0
```

ADDING A USER TO THE STUDENT DATABASE

Students can be added to the student database using the following syntax:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=add&DB=student
      &COURSE=<course_id>&AUTH=<32_byte_mac>&User%20ID=<user_id>
      &Password=<password>[&field1=<field1>][&fieldn=<fieldn>]
      [&ENCRYPTED=<1_or_0>]HTTP/1.0
```

where:

Key	Value	Description
COURSE	WebCT Course ID	The WebCT course to which the user will be added.
AUTH	32 byte MAC	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
User ID	User ID	The User ID of the user being added.
Password	Password	The password to be used for the user being added. The API does not enforce minimum password lengths.
Field1 … Fieldn (optional)	Data associated with the column specified as a key.	Any of the default columns within the global database (First Name, Last Name, Courses, Registered Courses) can be modified using the syntax ColumnName=Value. Administrator-created columns can also be modified using this method.
ENCRYPTED (optional)	1	Enables pre-encrypted password support. With the encrypted argument set to 1, you should pass passwords encrypted with the standard UNIX DES method when using this setting
	0 (default)	Disables pre-encrypted password support (default). In this mode, passwords should be submitted as clear

Example

Add a student to the student database of the course ENGL588. In addition, add data to a pre-existing column "StudentNumber" (This is a custom column created by the designer). Because this user is being added to the student database only, they are considered an "orphan user" until a WebCT ID is associated with this User ID:

```
GET  /webct/public/serve_webctdb?OPERATION=add&DB=student
      &COURSE=ENGL588&AUTH=EB1A09F0BB299C23E99A5978587F49C1
      &User%20ID=flounder&Password=an1mal&First%20Name=Kent
      &Last%20Name=Dorfman&StudentNumber=123456789 HTTP/1.0
```

UPDATING USERS

UPDATING A USER IN THE GLOBAL DATABASE

Updating users in the global database is very similar to adding users. The syntax is as follows:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=update&DB=global
      &COURSE=<placeholder>&AUTH=<32_byte_mac>&WebCT%20ID=<webct_id>
      [&FIELD1=<field1>][&FIELDN=<fieldn>][&ENCRYPTED=<1_or_0>]HTTP/1.0
```

where:

Key	Value	Description
COURSE	Any alphanumeric string	This is a required placeholder value. You may use any alphanumeric value, but ensure that you use it in the calculation of the MAC.
AUTH	32 byte MAC	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
WebCT ID	Existing WebCT ID	The WebCT ID of the user being added. WebCT IDs may contain alphanumeric strings, underscores, and periods.
Password (optional)	Password	The password to be used for the user being updated. The API does not enforce minimum password lengths.
Field1 … Fieldn (optional)	Data associated with the column specified as a key.	Any of the default columns within the global database (First Name, Last Name, Courses, Registered Courses) can be modified using the syntax ColumnName=Value. Administrator-created columns can also be modified using this method.
	DELETE	The "_DELETE_" keyword deletes the data from the field specified in the key and sets it to undefined.
ENCRYPTED (optional)	1	Enables pre-encrypted password support. With the encrypted argument set to 1, you should pass passwords encrypted with the standard UNIX DES method when using this setting
	0 (default)	Disables pre-encrypted password support (default). In this mode, passwords should be submitted as clear-text.

Notes:

- The *User Data* setting in the WebCT Administrator interface affects how updating the Courses column will modify the student database when unlinking WebCT IDs from User IDs. If the User Data setting is selected, user data is left in the student database.
- The Standard API always overwrites the Courses and Registered Course fields when updating. If you supply a Courses field in your update, the user's WebCT ID will be linked to the courses that you specify, and unlinked from any pre-existing courses that you do not specify.
- The Courses field uses a colon as a delimiter between courses, and a semicolon as a delimiter between user types. Thus the string "HKIN100;D:HKIN200;TA:HKIN300;S" indicates that a user is to be added to HKIN100 as a designer, HKIN200 as a teaching assistant and HKIN300 as a student. If no user type is specified, WebCT will default to adding the user as a student. Similarly, the Registered Courses field is colon delimited. See the *WebCT 3.7 Campus Edition System Administrator's Guide* for more information on the Courses and Registered courses field.

Example

A user is currently enrolled in three courses: ENGL101 as a designer, ENGL560 as a student, and ENGL477 as a teaching assistant. This example unlinks the WebCT ID from the User ID for ENGL 560 and ENGL 477, and adds the WebCT ID to the course ENGL101 as designer.

```
GET  /webct/public/serve_webctdb?OPERATION=update&DB=global&COURSE=xxxx&AUTH=EB1A
     09F0BB299C23E99A5978587F49C1&WebCT%20ID=pinto&Courses=ENGL101;D:ENGL101;D
     HTTP/1.0
```

The user is unlinked from the two courses because API updates always overwrite fields.

UPDATING A USER IN THE STUDENT DATABASE

Updating students in the student database is very similar to adding students. The syntax is as follows:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=update&DB=student
     &COURSE=<course_id>&AUTH=<32_byte_mac>&User%20ID=<user_id>
     [&field1=<field1>][&fieldn=<fieldn>][&ENCRYPTED=<1_or_0>]HTTP/1.0
```

where:

Key	Value	Description
COURSE	WebCT Course ID	The WebCT course in which the user's data is updated.
AUTH	32 byte MAC	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
User ID	User ID	The User ID of the user being added.
Password (optional)	Password	The password to be used for the user being added. The API does not enforce minimum password lengths.
Field1 … Fieldn (optional)	Data associated with the column specified as a key.	Any of the default columns within the global database (First Name, Last Name, Courses, Registered Courses) can be modified using the syntax ColumnName=Value. Administrator-created columns can also be modified using this method.
	DELETE	The "_DELETE_" keyword deletes the data from the field specified in the key and sets it to undefined.
ENCRYPTED (optional)	1	Enables pre-encrypted password support. With the encrypted argument set to 1, you should pass passwords encrypted with the standard UNIX DES method when using this setting.

Key	Value	Description
	0 (default)	Disables pre-encrypted password support (default). In this mode, passwords should be submitted as clear-text.

Example

In the following example, a student record is updated with information for the instructor-added numeric columns “Student Participation" and "Bonus" in the course MATH100.

```
GET webct/public/serve_webctdb?OPERATION=update&DB=student&
      COURSE=MATH100&AUTH=EB1A09F0BB299C23E99A5978587F49C1
      &User%20ID=otter&Student%20Participation=100&Bonus=34 HTTP/1.0
```

DELETING USERS

DELETING A USER FROM THE GLOBAL DATABASE

The syntax for deleting a user from the global database is as follows:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=delete&DB=global
      &COURSE=<placeholder>&AUTH=<32_byte_mac>&WebCT%20ID=<webct_id>
      HTTP/1.0
```

where:

Key	Value	Description
COURSE	Any alphanumeric string	This is a required placeholder value. You can use any alphanumeric value, but ensure that you use it in the calculation of the MAC.
AUTH	32 byte MAC	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
WebCT ID	WebCT ID	The WebCT ID of the user being deleted.

Note: The *User Data* setting in the WebCT Administrator interface affects whether user data is left in a course when a user record is deleted from the global database. If the *User Data* setting is selected, user data is left in the student database.

Example

In this example, the user record for the user with the WebCT ID “neidermeyer” is deleted from the global database:

```
GET  /webct/public/serve_webctdb?OPERATION=delete&DB=global&COURSE=xxxx
      &AUTH=EB1A09F0BB299C23E99A5978587F49C1&WebCT%20ID=neidermeyer
      HTTP/1.0
```

DELETING A USER FROM THE STUDENT DATABASE

The syntax for deleting a student from the student database is as follows:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=delete&DB=student
      &COURSE=<course_id>&AUTH=<32_byte_mac>&User%20ID=<user_id>
      HTTP/1.0
```

where:

Key	Value	Description
COURSE	WebCT Course ID	The WebCT course from which the user will be deleted.
AUTH	32 byte MAC	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
User ID	User ID	The User ID of the user being deleted.

Example

In this example, the student with the User ID "stork" is deleted from the course PSYCH204-23:

```
GET  /webct/public/serve_webctdb?OPERATION=delete&DB=student
      &COURSE=PSYCH204-23&AUTH=EB1A09F0BB299C23E99A5978587F49C1
      &User%20ID=stork HTTP/1.0
```

FINDING USERS

FINDING A USER IN THE GLOBAL DATABASE

To find a user's global database record, the following syntax is used:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=find&DB=global
      &COURSE=<placeholder>&AUTH=<32_byte_mac>&WebCT%20ID=<webct_id>
      [USER%20TYPE=<1_or_0>]HTTP/1.0
```

where:

Key	Value	Description
OPERATION	find	Finds the user record for a given WebCT ID
COURSE	Any alphanumeric string	This is a required placeholder value. You can use any alphanumeric value, but ensure that you use it in the calculation of the MAC.
AUTH	32_byte_mac	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
WebCT ID	WebCT_ID	The WebCT ID of the record you want to display.

Key	Value	Description
USER TYPE (optional)	1	With the User Type option enabled, the global database record generated includes user type information that indicates whether a user is a designer, student, or teaching assistant for the course.
	0 (default)	No user type information is generated.

Example

In this example, the complete record including user type information is returned for the user with the WebCT ID "pinto", who is enrolled in three courses.

```
GET  /webct/public/serve_webctdb?OPERATION=find&DB=global&COURSE=xxxx
      &AUTH=EB1A09F0BB299C23E99A5978587F49C1&WebCT%20ID=pinto
      &USER%20TYPE=1 HTTP/1.0
```

The Web server returns the following, not including http headers:

```
Success: WebCT ID=pinto,First Name=Larry,Last Name=Kroger,Courses=
      ENGL100;D:ENGL560;S:ENGL477;TA
```

FINDING A USER IN THE STUDENT DATABASE

To find a student's student database record, the following syntax is used:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=find&DB=student
      &COURSE=<course_id>&AUTH=<32_byte_mac>&User%20ID=<user_id> HTTP/1.0
```

where:

Key	Value	Description
OPERATION	find	Finds a user's record from a WebCT ID.
COURSE	Any alphanumeric string	The course that you are searching.
AUTH	32_byte_mac	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
User ID	User ID	The User ID of the record you wish to display.

Example

In this example, a complete student database record is displayed for the user with User ID "chip" in the course "HKIN455":

```
GET  /webct/public/serve_webctdb?OPERATION=find&DB=student
      &COURSE=HKIN455=AUTH=EB1A09F0BB299C23E99A5978587F49C1&User%20ID=chip
      HTTP/1.0
```

The Web server returns the following, not including http headers:

```
Success: First Name=Chip,Last Name=Diller,User ID=chip,Quiz1=36,Assignment1=10
```

CHANGING WEBCT IDS

CHANGING A USER'S WEBCT ID

To change a WebCT ID for a user, use the syntax:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=changeid&DB=global
      &COURSE=<placeholder>&AUTH=<32_byte_mac>&Old%20ID=<old_webct_id>
      &New%20ID=<new_webct_id> HTTP/1.0
```

where:

Key	Value	Description
OPERATION	changid	Changes the WebCT ID of a user
COURSE	Any alphanumeric string	This is a required placeholder value. You can use any alphanumeric value, but ensure that you use it in the calculation of the MAC.
AUTH	32_byte_mac	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
Old ID	Old WebCT ID	The WebCT ID of the record you want to change.
New ID	New WebCT ID	The WebCT ID that you want to assign to the user.

Example

In this example, the WebCT ID "flounder" is changed to "dorfmank":

```
GET  /webct/public/serve_webctdb?OPERATION=changeid&DB=global&COURSE=xxxx
      &AUTH=EB1A09F0BB299C23E99A5978587F49C1&Old%20ID=flounder
      &New%20ID=dorfmank HTTP/1.0
```

The Web server returns the following, not including http headers:

```
Success:
```

SECTION 2: CAMPUS EDITION

CHAPTER 1
USER AUTHENTICATION

WebCT 3.7 Campus Edition (CE) provides two major methods for user authentication:

Browser Based Authentication	• Users are authenticated through a browser dialog box that prompts for a username and password. The username and password are verified against WebCT internal databases. If the user is authorized, a Basic Authentication Header is provided. Subsequent page accesses to WebCT are authorized according to the browser header. • This authentication method is used in previous versions of WebCT.
Ticket Based Authentication	• Users are authenticated through a logon page that prompts for a username and password. The username and password are verified against either WebCT internal databases or against an external password database. If the user is authenticated, the user is issued a browser cookie that serves as a ticket. Subsequent page accesses to WebCT are authorized according to the ticket. • Institutions that choose ticket-based authentication have the option of implementing automatic signon to WebCT. With this feature implemented, institutions that have portal solutions or other secure environments can create a seamless environment for users by pre-authenticating them into WebCT.

CHOOSING AN AUTHENTICATION METHOD

Many of the features of WebCT 3.7 CE require ticket-based authentication, including:

External password database authentication using LDAP, Kerberos, Windows 2000 Domain Controller, or a custom implementation

- Logout
- Server lockdown
- Automatic signon
- Session timeout
- Customizable logon page

Browser based authentication is primarily provided in WebCT 3.7 as a legacy option. Choose this method of authentication if:

- Your institution has an information technology policy forbidding the use of applications that employ browser cookies.
- It is critical that the user interface of WebCT remain the same as previous versions.

Browser Based Authentication Process

Browser-based authentication has served as the standard authentication method for all previous versions of WebCT. When a user a requests a URL, authentication of the user occurs as follows:

1. The Web server checks to see if the requested URL requires authorization.
2. If none is required (e.g., a user requests a course Welcome Page), then the Web server delivers the page to the browser.
3. If authorization is required, the Web server checks to see if the user has already provided a username and password by checking to see if a valid Basic Authentication Header was provided in the request. If the header is valid, the page is delivered.
4. If the Basic Authentication Header is invalid, or no header is provided, the user is prompted with a username and password dialog box. The cycle is then repeated.

Ticket Based Authentication Process

When a user requests a URL, authentication of the user occurs as follows:

1. The Web server checks to see if the requested URL requires authorization.
2. If none is required, the Web server delivers the page to the browser.
3. If authorization is required, WebCT checks for a valid ticket.
4. If a valid ticket is found (i.e. the user has been authenticated and is authorized for the resource), the page is delivered to the browser.
5. If a ticket is not found, a logon form is delivered to the browser. The user submits the form and WebCT authenticates the user, issuing their browser a cookie. The URL is re-requested and the cycle repeats.

How WebCT Generates Tickets

WebCT tickets (in the form of browser cookies) contain the following information:

- Username
- Encrypted Password (DES encryption)
- Timestamp (UNIX Epoch format)
- Message Authentication Code (MAC)

The MAC is generated in three steps:

1. The username, encrypted password, timestamp, user agent information (if sent), and a shared secret value are concatenated.
2. The concatenated string is encrypted with the MD5 algorithm.
3. The encrypted string is encrypted a second time with the MD5 algorithm.

IMPLEMENTING TICKET BASED AUTHENTICATION

With ticket-based authentication, you can use one or more authentication sources. WebCT supports the following authentication sources:

- WebCT's internal database (default)
- LDAP
- Kerberos
- Windows 2000 Domain Controller
- a custom authentication source.

CHOOSING AN AUTHENTICATION SOURCE

The authentication source(s) that you choose should be based on what your institution has already implemented. If your institution is using a centralized password management or single signon solution that is not directly supported, you may want to consider a custom implementation using WebCT's open source authentication code, written in C. For more information on custom authentication, see the section *Implementing Custom Authentication*.

The following table describes each type of authentication source.

WebCT Internal Database	• This is the best option for institutions that do not have a single signon solution. • This is the easiest option to deploy as there are no external systems to manage.
LDAP	• This is the open standard for providing directory services such as email addresses, telephone numbers, addresses, etc. to the Internet. • Many institutions have discovered that LDAP can also serve as an authentication database as part of a single signon environment. • LDAP is not a true authentication source, so it lacks many of the features seen in purpose-built authentication sources.
Kerberos/Windows 2000 Domain Controller	• Kerberos is an authentication system that enables two parties to exchange private information across a network. A unique key, called a ticket, is assigned to each user who logs on to the network.
Custom Authentication	• If your institution uses a single signon solution that is not directly supported, (e.g. Radius, IMAP), your institution can modify WebCT's open source authentication code by using the WebCT Open Authentication Kit. • To obtain the WebCT Open Authentication Kit (WOAK), contact your account representative. • You will need an experienced C programmer to write the authentication function.

USING ONE AUTHENTICATION SOURCE

1. From the *Admin* toolbar, click **Server Mgmt**. The *Server Mgmt* toolbar appears.
2. From the *Server Mgmt* toolbar, click **Settings**. The *Administrator Settings* screen appears.
3. Under *User Authentication*, select *Use ticket based authentication*.

4. Choose whether the **Logout** link should appear in the course *Menu Bar*:
    - To display the **Logout** link, select *Display Logout link in course Menu Bar*.
    - To hide the **Logout** link, deselect *Display Logout link in course Menu Bar*. **Note:** If you run WebCT in a framed environment (such as a portal) where a logout link or "Return to Portal" link already exists, you may prefer to hide the **Logout** link.
5. In the *Ticket shared secret value* text box, either leave the shared secret value that was automatically generated by WebCT or enter a new shared secret value. For security reasons, the default value "secret" does not work. The secret value
    - is case-sensitive
    - cannot exceed 256 characters
    - cannot contain tab or other control characters
    - should not contain end-of-line characters. **Note:** By default, the UNIX text editors vi and pico automatically add end-of-line characters. Check the file size to ensure that the number of characters equals the number of bytes.
6. In the *Tickets remain valid for* text box, enter the number of minutes until ticket time-out. This value controls the expiry time of the ticket based on the user's last access and therefore affects how long a user can stay logged on while inactive. The default is 180 minutes.
7. Choose whether to allow WebCT authentication across a domain. Authentication across a domain allows users to access all servers in the domain, without having to re-authenticate for each one.
    - To allow authentication across a domain:
        c) Select *Allow WebCT authentication across a domain*.
        d) In the *Please specify your domain* text box, enter the domain name. The domain name must have a period in front of it. Example: `.webct.com`
    - To disallow authentication across a domain, select *Do not allow WebCT authentication across a domain*.
8. Under *User is authenticated using*, from the drop-down list for the authentication source that you are using, select *First*.
9. For all other authentication sources, select *Never*.
10. Scroll to the bottom of the screen and click **Update**.

USING MULTIPLE AUTHENTICATION SOURCES

You can integrate third-party authentication sources, such as LDAP, Kerberos, or a custom authentication source with WebCT. For example, use multiple authentication sources if your institution requires a failover authentication scheme to authenticate users who do not have an account in the primary authentication database. Users who are not authenticated by the primary authentication source can be authenticated by secondary sources, such as the internal WebCT database.

1. From the *Admin* toolbar, click **Server Mgmt**. The *Server Mgmt* toolbar appears.
2. From the *Server Mgmt* toolbar, click **Settings**. The *Administrator Settings* screen appears.
3. Under *User Authentication*, select *Use ticket based authentication*.
4. Choose whether the **Logout** link should appear in the course *Menu Bar*:
    - To display the **Logout** link, select *Display Logout link* in course Menu Bar.

- To hide the **Logout** link, deselect *Display Logout link in course Menu Bar*. **Note:** If you run WebCT in a framed environment (such as a portal) where a logout link or "Return to Portal" link already exists, you may prefer to hide the **Logout** link.

5. In the *Ticket shared secret value* text box, either leave the shared secret value that was automatically generated by WebCT or enter a new shared secret value. For security reasons, the default value "secret" does not work. The secret value

   - is case-sensitive
   - cannot exceed 256 characters
   - cannot contain tab or other control characters
   - should contain end-of-line characters. **Note:** By default, the UNIX text editors vi and pico automatically add end-of-line characters. Check the file size to ensure that the number of characters equals the number of bytes.

6. In the *Tickets remain valid* for text box, enter the number of minutes until ticket time-out. This value controls the expiry time of the ticket based on the user's last access and therefore affects how long a user can stay logged on while inactive. The default is 180 minutes.

7. Choose whether to allow WebCT authentication across a domain. Authentication across a domain allows users to access all servers in the domain, without having to re-authenticate for each one.

   - To allow authentication across a domain:
     a) Select *Allow WebCT authentication across a domain*.
     b) In the *Please specify your domain* text box, enter the domain name. The domain name must have a period in front of it. Example: `.webct.com`
   - To disallow authentication across a domain, select *Do not allow WebCT authentication across a domain*.

8. Under *User is authenticated using*, specify when to use the authentication source(s):

   - If you are using the internal WebCT password database, from the corresponding drop-down list, select when it should be used in the authentication sequence. **Important:** If you are using the internal WebCT database in a failover authentication scheme, it is strongly recommended that you
     - use the WebCT database last in the authentication sequence.
     - do not use passwords that can be guessed (for example: *webct* or *password*).
   - If you are using LDAP:
     a) From the LDAP server drop-down list, select when it should be used in the authentication sequence.
     b) Specify the LDAP settings. (See the *Specifying the LDAP Settings* section in this guide).
   - If you are using Kerberos or Windows 2000 Domain Controller:
     a) From the *MIT Kerberos V5 KDC* or *Windows 2000 Domain Controller* drop-down list, select when it should be used in the authentication sequence.
     b) Specify the Kerberos settings or Windows 2000 Domain Controller settings (See the *Specifying the Kerberos Settings* section in this guide). If you are using a custom authentication source, from the corresponding drop-down list, select when it should be used in the authentication sequence.

9. Scroll to the bottom of the screen and click **Update**.

SPECIFYING THE LDAP SETTINGS

1. Under *LDAP settings*, in the *LDAP Server Name* text box, enter the name of your LDAP server.
2. In the *LDAP Port* text box, enter the port of your LDAP server.

3. In the *Base DN* text box, enter the top level of the LDAP directory tree where your WebCT user records are stored. This directs the authentication program to search in the appropriate directory on your LDAP server.
4. In the *WebCT ID Attribute* text box, enter the attribute or field of the user record where the WebCT ID is stored.
5. In the *Manager DN* text box, enter the LDAP server manager's distinguished name.
6. In the *Manager Password* text box, enter the LDAP server manager's password.
7. Click **Update**.

Important: If you are using LDAP in a multiple authentication scheme, you must also specify the sequence in which it should be used.

SPECIFYING THE KERBEROS SETTINGS

Note:

- Unix users: Kerberos requires a properly configured krb5.conf file in the /etc directory.
- Windows users: Kerberos requires a properly configured krb5.ini file in the <webct_install_dir>\webct\webct\generic\ticket folder.

1. Under *Kerberos/Domain Controller* settings, in the *Realm/Domain Name* text box, enter the Kerberos Realm name. **Note:** Each entry in the KDC is called a principal and has the format:
   `username/instance@Kerberos Realm`
   Example: `johnsmith/admin@MYINSTITUTE.EDU`
   In this example, the Realm is `MYINSTITUTE.EDU`.
2. In the *Instance* text box, enter the Kerberos Instance name. In the example above, the Instance is `admin`.
3. Click **Update**.

Important: If you are using Kerberos in a multiple authentication scheme, you must also specify the sequence in which it should be used.

SPECIFYING THE WINDOWS 2000 DOMAIN CONTROLLER SETTINGS

1. Under *Kerberos/Domain Controller* settings, in the *Realm/Domain Name* text box, enter the Windows domain name.
2. Leave the *Instance* text box empty.
3. Click **Update**.

Important: If you are using *Windows 2000 Domain Controller* in a multiple authentication scheme, you must also specify the sequence in which it should be used.

IMPLEMENTING CUSTOM AUTHENTICATION

For institutions that use external password databases that are not directly supported (e.g. Radius, IMAP), WebCT allows modification of the WebCT open source authentication code by using the WebCT Open Authentication Kit (WOAK). To obtain the WebCT Open Authentication Kit for your operating system, contact your account representative. In addition, you will need an experienced C programmer to write the authentication function.

UNIX/LINUX

To compile the WOAK, ensure that you have the Free Software Foundation's GCC compiler installed (http://www.gnu.org/software/gcc/). Other C compilers are not recommended.

It is beyond the scope of this document to describe an exact procedure for code development. You should follow basic rules such as not developing on live servers, make appropriate backups of important files, and do as much testing as possible with your custom code. The following is provided as a general guide:

1. Make a backup copy of your original ticketLogin executable in the `[install_dir]/webct/webct/generic/ticket/` directory.
2. Extract the WebCT Open Authentication Kit to a working directory.
3. In the `[woak_directory]/compile` directory of the WOAK, modify the Makefile with a text editor so that your system architecture is uncommented (e.g. A Solaris developer should uncomment the `configure/solaris-sparc` line and make sure that both the `configure/linux-libc6` and `configure/aix lines` are commented out.) The Makefile is set up for Linux systems by default.
4. Open the file `custom_auth.c` within the `[woak_directory]/ticket` directory.
5. Modify the function `user_is_authentic_other` so that it returns 1 if the username and password passed to the function is authentic or –1 if it is not.
6. Make any changes necessary to your Makefile in order for it to compile with your code.
7. Compile the ticketLogin executable by issuing the `make` command within the `custom_auth/compile` directory.
8. Copy the new ticketLogin executable to your test server.

WINDOWS NT/2000

To compile the WebCT Open Authentication Kit (WOAK), ensure that you have Microsoft Visual Studio 6 (http://msdn.microsoft.com/vstudio) installed. Other C compilers are not recommended.

It is beyond the scope of this document to describe an exact procedure for code development. The following is provided as a general guide.

1. Make a backup copy of your original ticketLogin executable in the `[install_dir]/webct/webct/generic/ticket/` directory.
2. Extract the WOAK into a working directory.
3. Open up the WOAK project within Visual Studio 6.
4. In the file custom_auth.c, implement the function user_is_authentic_other so that it returns AUTH_DECLINED if the user does not exist in the password database, AUTH_VALID if the password is valid, and AUTH_FAILED if the password is not valid.
5. Compile ticketLogin and copy the executable to your test server.

CHAPTER 2
AUTOMATIC SIGNON FROM OTHER SYSTEMS

Automatic signon allows institutions to create seamless computing environments. Users can move from an application where they are authenticated to WebCT without retyping usernames and passwords. For example, Automatic signon could allow users to log on to their campus portal, browse campus events, and then click a link to their WebCT course, upon which they are automatically logged on, without being prompted for a username or password.

AUTOMATIC SIGNON PROCESS

The automatic signon process will vary depending on the type of system with which you are integrating WebCT. For an institution that has a campus portal, the process may occur as follows:

1. A user accesses their campus portal account, using their portal username and password.
2. The portal obtains the user's *myWebCT* and/or WebCT course information either by
   - obtaining the information from an external source, such as a student information system.
   - executing a local program that makes a Standard API call to obtain the information. The local program can use one of the following API commands:
     - Standard API command `homearea_xml`, which uses the WebCT ID and server base address to export a user's *myWebCT* in XML format (see *Chapter 5: Standard API*)
     - Standard API command `find`, which uses the WebCT ID to find a user's global database record (see *Chapter 5: Standard API*)
3. The user clicks a link to the WebCT server (either to their *myWebCT* or directly into a course).
4. The portal executes a local program that makes an IMS API call to find the user's IMS ID and IMS source.
5. WebCT returns the IMS ID and IMS source.
6. Optional: The portal stores the IMS ID and IMS source locally with the user's record so that subsequent requests to the WebCT server are faster.
7. The portal executes a local program that creates a Message Authentication Code (MAC) from the data (the IMS ID and IMS source, a timestamp, and a destination URL) and the shared secret value. The local program assembles the data and MAC into an http request and then sends the http request via the user's browser.
8. WebCT verifies the validity of the http request and issues a ticket in the form of a browser cookie.
9. The user is redirected to the URL provided in the request (e.g., the course *Homepage* or their *myWebCT*).

IMPLEMENTING AUTOMATIC SIGNON

Implementing automatic signon involves two steps:

1. Setting shared secret values and enabling ticket based authentication
2. Developing a program to automatically authenticate a user

Step 1. can be accomplished by a WebCT administrator who has basic knowledge of the WebCT file system. Step 2 requires an experienced Web developer.

1. SETTING SHARED SECRET VALUES AND ENABLING TICKET BASED AUTHENTICATION

Shared secret values are key security components for Automatic signon as they are used for authenticating messages from external servers. Implementing Automatic signon requires setting two shared secret values, which ensure that only messages from trusted servers are processed:

- the Automatic signon secret
- the API secret

First, set the shared secret value for Automatic signon:

1. Using a text editor, open the file `<webct_install_dir>/webct/webct/generic/autosignon/autosignon_secret`
2. Change the first line of the file to your desired secret. (For security reasons, the default value "SECRET" does not work).
   - It cannot exceed 256 characters.
   - It cannot contain tab or other control characters.
   - It should not contain end-of-line characters. **Note**: By default, the UNIX text editor vi and pico automatically add end-of-line characters. Check the file size to ensure that the number of characters equals the number of bytes.
   - It is case-sensitive
3. Change the first line of the file to your desired secret. (For security reasons, the default value "SECRET" does not work).Save the file.

Now, set the API shared secret value:

4. Open the file `<webct_install_dir>/webct/webct/generic/api/api_secret`
5. Change the first line of the file to your desired secret, following the guidelines in step 2.
6. Save the file.

Now, logon to the Administrator interface, and enable ticket-based authentication:

7. From the *Admin* toolbar, click **Server Mgmt**. The *Server Mgmt* toolbar appears.
8. From the *Server Mgmt* toolbar, click **Settings**. The *Administrator Settings* screen appears.

9. Under *User Authentication*, select *Use ticket based authentication.*

2. Developing a Program to Automatically Authenticate a User

The most substantive part of implementing automatic signon is developing a program that automatically authenticates users into WebCT. The program must:

- find a user's IMS ID and IMS source via the IMS API
- make a request to the Automatic Signon CGI

Each of these requests requires the creation of a MAC to ensure the authenticity of the request.

Creating Message Authentication Codes

Because the Web interfaces to the Standard API and Automatic signon reside in public directories, Message Authentication Codes (MACs) are required to ensure that only messages from trusted servers are processed.

WebCT provides three options to assist you in creating MACs:

1. A C function that you may integrate and compile into your C program
2. An executable file to which you make a system call from your program
3. Instructions for generating a MAC using a language of your choice

Option 1: Using the get_authentication C function

The get_authentication function generates a MAC from an array of data and a shared secret value.

The source code necessary to use the C function is located in `<webct_install_directory>/webct/webct/generic/api/security/`

A test program, which contains a Makefile for UNIX based systems, is also provided.

The file `api_security.c` contains the `get_authentication` function.

get_authentication	Generates a MAC from an array of data and a shared secret value
Syntax	char* get_authentication (int i, char* data[], char* secret, char* encrypted_data)
Returns	32 byte alphanumeric MAC
Parameter	**Description**
`i`	The number of elements in the array data.
`data`	Array of all values to be used in generating the MAC. The data should not be URL encoded.
`secret`	The shared secret value.
`encrypted_data`	The memory location of the MAC. It must be at least 32 bytes

long.

OPTION 2: USING THE THE MESSAGE AUTHENTICATION CODE GENERATOR (GET_MD5 EXECUTABLE)

The Message Authentication Code (MAC) generator generates a MAC from a shared secret value and a string consisting of the IMS ID, a timestamp, and a destination URL.

Use the Message Authentication Code (MAC) generator (an executable called get_md5) if you are not working in the C language, or do not want to create a function to create the MAC. You can make a system call to get_md5 from your program and have the authentication string returned. The get_md5 executable has no dependencies on WebCT and can be copied to other servers as required. If you need a get_md5 executable for an operating system other than the one your WebCT server is running on, you can download several pre-compiled binaries for other operating systems from `http://download.webct.com`

get_md5	Generates a MAC from a shared secret value and a string to be encrypted (consisting of the IMS ID, a timestamp, and a destination URL).
Syntax	get_md5 <shared_secret_filename> <string_to_encrypt>
Returns	32 byte alphanumeric MAC

Attribute	Description
`shared_secret_filename`	The filename where the shared secret value is stored.
`data_to_encrypt`	The data to be encrypted. Data should not be URL encoded.

An example of using the get_md5 program to generate a MAC from a shared secret value and the data string described above follows:

eg. MAC = get_md5 api_secret 2A508D8EB5EB2D596DD937E2B8835100 982187291 http://webct.institution.edu:8900/ SCRIPT/ENGL100-001/scripts/serve_home

OPTION 3: CREATE A MAC USING A LANGUAGE OF YOUR OWN CHOICE

If you want to create MACs within your code, (e.g. you are writing your code in Java and don't want to make a system call), you can create a MAC with the following procedure:

1. Calculate the total of the ASCII values of all the characters in the passed data.
2. Convert the total of the ASCII values into a string.
3. Append the shared secret value.
4. Encrypt the string into a 16-byte string using the MD5 algorithm.
5. Convert the 16-byte string into a 32-byte alphanumeric string to make it URL-friendly.

FINDING THE IMS ID FOR AUTOMATIC SIGNON

To send a request to Autosignon, the program must first find a user's IMS ID and IMS source. The program can find the IMS ID and IMS source by making an IMS API call using the `get_person_ims_info` operation. Optionally, after finding the IMS ID and IMS source, the program can store the IMS ID and IMS source locally so that the next time they are required, the program can read them locally and then pass them to the Automatic signon CGI without having to make an API call.

FINDING THE IMS ID USING THE WEB-BASED IMS API

The syntax for a Web-based request to the IMS API to find an IMS ID is as follows:

```
<GET | POST>  /webct/ims/serve_ep_api.pl?ACTION=configure&OPTION=get_person_ims_i
      nfo&GLOBALID=<WEBCTID>&TIMESTAMP=<unix_epoch_time>
      &AUTH=<32_byte_mac>
```

OPTION	get_person_ims_info	Finds a user's IMS ID and IMS source from a WebCT ID
GLOBALID	An existing WebCT ID	An existing WebCT ID is required when using the `get_person_ims_info` option
TIMESTAMP	UNIX epoch timestamp	Time stamp in UNIX epoch format (seconds since midnight GMT, Jan 1, 1970)
AUTH	A valid MAC	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.

When developing a program for a Web-based API, you should keep the following points in mind:

- You can use either GET or POST methods to submit requests
- Requests must be URL encoded (e.g. spaces should be replaced with %20)
- Key/value pairs can be separated by ampersands (&) or plus signs (+)
- Key/value pairs may appear in any order

FINDING THE IMS ID USING THE COMMAND LINE IMS API

The IMS API executable ep_api.pl is in the following directory:
`<webct_install_dir>/webct/generic/ims`

The syntax for a command line request to the IMS API to find an IMS ID is as follows:

```
ep_api.pl configure get_person_ims_info <WEBCTID>
```

where:

Argument	Input	Description
WEBCTID	An existing WebCT ID	A WebCT ID is required when using the get_person_ims_info option.

Example

Find the IMS ID for the WebCT ID jdoe:

```
ep_api.pl configure get_person_ims_info jdoe
```

The IMS API returns:

```
Success:

IMS id=2A508D8EB5EB2D596DD937E2B8835100

IMS source=WebCT
```

FINDING WUUIS

FINDING THE WUUI USING THE WEB-BASED STANDARD API

Important: Since the release of WebCT 3.6, the use of the `WebCT Unique Universal Identifier` (WUUI) for Automatic Signon and the `find_wuui` operation are deprecated. With WebCT moving towards the use of the IMS specifications, which are becoming standards in the learning community, the IMS ID and IMS source are now preferred over the WUUI. Although the use of the WUUI is deprecated, the functionality will still be supported for 3.7.

The syntax for a Web-based request to the Standard API is as follows:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=<operation>&DB=global
      &field1=<field1>&COURSE=<placeholder>&AUTH=<32_byte_mac>
      HTTP/1.0
```

Key	Value	Notes
`OPERATION`	`find_wuui`	Finds a user's WUUI for a given a WebCT ID (**Note**: deprecated; not recommended for use)
	`find_ims_id_wuui`	Finds a user's WUUI for a given an IMS ID
`DB`	`global`	Although this value may also be `student`, for the application of finding WUUIs, you will always use `global`
`field1`	`WebCT_ID`	Use if the operation is find_wuui
	`IMS_ID`	Use if the operation is find_ims_id_wuui
`COURSE`	`Any alphanumeric string`	This is a generic placeholder value. You can use any value, but ensure that you use it in the calculation of the MAC.
`AUTH`	`32 byte MAC`	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.

FIND A USER'S WUUI FROM AN IMS ID

Find the WUUI for the IMS ID (Person→SourcedID→ID) "123456789":

```
GET  /webct/public/serve_webctdb?OPERATION=find_ims_id_wuui
     &DB=global&IMS%20ID=123456789&COURSE=xxxx
     &AUTH=EB1A09F0BB299C23E99A5978587F49C1 HTTP/1.0
```

The Web server returns the following, not including http headers:

```
Success: #WUUI = 6321BB2537BE7F1E26375D4E1687EE1F
```

FINDING THE WUUI USING THE COMMAND LINE STANDARD API

Important: Since the release of WebCT 3.6, the use of the WUUI for Automatic Signon and the `find_wuui` operation are deprecated. With WebCT moving towards the use of the IMS specifications, which are becoming standards in the learning community, the IMS ID and IMS source are now preferred over the WUUI. . Although the use of the WUUI is deprecated, the functionality will still be supported for 3.7.

The Standard API executable webctdb is in the following directory:
`<webct_install_dir>/webct/webct/generic/api`

The general syntax using the command line Standard API to find WUUIs is as follows:

```
webctdb <find_ims_id_wuui> global xxxx <WEBCTID | IMSID>
```

Where:

- `find_ims_id_wuui` is the operation to find a WUUI using an IMS ID
- `global` is the name of the database you are accessing
- `xxxx` is a required placeholder
- `WEBCTID` is the WebCT ID of the user whose WUUI you are trying to find
- `IMSID` is the IMS ID of the user whose WUUI you are trying to find

MAKING A REQUEST TO THE AUTOMATIC SIGNON CGI

Once the program has determined the IMS ID and IMS source, it must pass the IMS ID and IMS source to the Autosignon CGI, which then logs the user on to WebCT.

The general syntax for an Automatic signon request is as follows:

```
http://<webctserver>:<port>/webct/public/autosignon?IMS%20id=<IMS id>
     &Time%20Stamp=<unix_epoch_time>&URL=<url>&MAC=<32_byte_mac>
```

Key	Value	Notes
`IMS id`	An IMS id	This is the IMS ID that has either been found using an IMS API call or is stored locally.
`Time Stamp`	unix_epoch_time	Time stamp in UNIX epoch format (seconds since midnight GMT, Jan 1, 1970).

URL	URL	The destination URL.
MAC	32 byte mac	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.

The following example shows a user being logged onto the course *Homepage* of the course ID ENGL100-001:

```
http://webct.institution.edu:8900/webct/public/autosignon?
      IMS%20id=2A508D8EB5EB2D596DD937E2B8835100&Time%20Stamp=982187291
      &URL=http://webct.institution.edu:8900/SCRIPT/ENGL100-
      001/scripts/serve_home&MAC=0A0D776506D70AE16537560CBDE4EE1
```

The server responds by issuing the browser a cookie and sending the user to the URL, in this case the homepage for ENGL100-001.

CHAPTER 3
OVERVIEW OF THE APPLICATION PROGRAMMING INTERFACES

Application Programming Interfaces (APIs) allow users and other systems to directly interface with WebCT without the graphical user interface. WebCT 3.7 CE provides two APIs, the proprietary Standard API, and the IMS Enterprise API. The IMS Enterprise API is exclusive to WebCT 3.7 Campus Edition and is compliant with the IMS Global Learning Consortium, Inc. (http://www.imsproject.org) enterprise API specification. Both APIs have two interfaces, a command line interface and a Web-based interface.

DIFFERENCES BETWEEN THE IMS ENTERPRISE API AND THE STANDARD API

Some functions are available in both the IMS API and Standard API. The following table summarizes the functionality of each API.

FUNCTIONAL DIFFERENCES

Function				
	Command Line	Web-Based	Command Line	Web-Based
Export Midterm/Final Grades from Course	√	√		
Add/Update/Delete Multiple Courses	√	√		
Add/Update/Delete Single Users				
from/to global database	√	√	√	√
from/to the student database [1] (Manage Students table in courses)			√	√
Add/Update/Delete Multiple Users				
from/to global database	√	√	√	
from/to the student database [1] (Manage Students table in courses)			√	
Set IMS IDs and IMS sources for users and courses	√	√		
Find Users	√	√	√	√
Find WUUIs				
Find IMS IDs	√	√		
Change WebCT IDs			√	√

[1] Indirectly, the IMS API can modify the student database when importing membership objects. However, direct modification is only available via the Standard API.

Export *myWebCT* in XML format			√	

OPERATIONAL DIFFERENCES

The IMS API relies on the exchange of XML data files to perform its functions. The Standard API relies on a variety of operations to carry out its functions, including the processing of delimited text files for batch operations, entering command line statements for single operations, and creating http query-strings for Web-based operations.

For features unique to each API, the choice of which to use is obvious. However, for managing user accounts in WebCT, the choice is more difficult since there is an overlap in functionality. In general, whenever your goal is to communicate with an IMS-compliant application, you should use the IMS API. Some advantages and disadvantages of each API for user management are summarized below:

	Advantages	Disadvantages
IMS API	• Allows you to set IMS IDs and IMS sources within WebCT • Facilitates two way communication between IMS-compliant systems (which rely on the IMS ID and IMS source)	• XML files must be created
Standard API	• Allows the modification of administrator-created columns • Can be simpler to use for user management since it is command driven and does not require XML files	• Difficult to integrate with IMS-compliant systems

CHOOSING THE APPROPRIATE INTERFACE FOR YOUR REQUIREMENTS

WebCT provides two interfaces to each of its APIs: a command line interface, and a Web-based interface. Choosing an interface is not a one-time decision; it will vary depending on the task that you need to accomplish. For example, if you want to process real-time updates from your institution's Student Information System (SIS), the Web-based interface may be more suitable. However, if you have to debug a user's problem, using the command line interface may be more suitable.

Use the following table as a guide for choosing the best interface.

Task/ Situation	Suggested Interface
Processing multiple records simultaneously (e.g. you want to populate the global database based on a batch extract from your institution's SIS)	command line
Processing a single record	command line
Integrating systems that are on the same physical server and run as the same user as WebCT	command line

Task/ Situation	Suggested Interface
Debugging	command line
Integrating external system with WebCT (e.g., you want to integrate your institution's SIS with WebCT)	Web-based

CHAPTER 4
IMS ENTERPRISE API

The IMS Enterprise API in WebCT 3.7 Campus Edition complies with version 1.01 of the IMS Enterprise Specification (available at http://www.imsproject.org/enterprise), the latest version of the specification at the time of release. The IMS API allows you streamline integration of WebCT with other products that also conform to the IMS specification.

FUNCTIONALITY IN THE IMS ENTERPRISE API

The IMS API has functionality that can be divided into three basic categories.

Import	• Create, modify, and delete course instances (group objects in IMS terminology) • Create, modify, and delete users (person objects in IMS terminology) • Register and deregister users from courses (associate person with group objects using the membership object in IMS terminology)
Export	• Create an XML file of all user, course, and course membership information • Create an XML file containing basic information for a single student record • Create an XML file containing a list of users with midterm and/or final grades for a given course
Configure	• Add or update the IMS source for a single course • Add or update the IMS ID for a single course • Add or update the IMS source for a single user • Add or update the IMS ID for a single user • Find the IMS ID and IMS source for a single user

TERMINOLOGY

The IMS Enterprise Information Model describes data structures that are used to provide interoperability of instructional management systems like WebCT with other enterprise systems. The information model defines several data objects, and WebCT maps the appropriate internal data to these objects. You should become familiar with the IMS data objects and their relation to WebCT. The following table summarizes some of the important terminology relevant to WebCT.

Important IMS Terminology for WebCT users

Term	Definition
Group Object	An object describing a group of users, most commonly a course instance. WebCT matches data in this object with data associated with courses such as the Course ID and Course Description.
Person Object	An object describing an individual. IMS data elements map to WebCT elements as follows: User ID maps to WebCT ID; Family maps to Last Name; and Given maps to First Name.
Membership Object	An object describing the membership of a person or group within a group. WebCT uses the membership object to modify the Courses field within the global database and to add instructors, students, and teaching assistants to their appropriate course databases.
Properties Object	An object containing general bookkeeping information for an IMS-compliant XML file. WebCT has no equivalent to the properties object.
IMS ID	The IMS ID is a unique identifier for an IMS object. All group, person, and membership objects have an associated IMS ID. Within an IMS-compliant XML file, the IMS ID refers to the <object>→SourcedID→ID
IMS source	The IMS source identifies the organization or system that assigned the IMS ID to the object. Group, person, and membership objects all have IMS sources Within an IMS-compliant XML file, the IMS source refers to the <object>→SourcedID→Source.

IMPLEMENTING THE IMS API

To implement the IMS API, consider the tasks that you need to perform and the systems that you want WebCT to interact with. A typical implementation of the IMS API with a Student Information System (SIS) might include the following steps.

1. Creation of an XML extract from a SIS that has group, person, and member data objects within it.
2. Bulk population of the WebCT global database using the Command Line IMS API to import the XML extract. This creates the courses, creates the users, and assigns users to courses.
3. Setup of an interface between the SIS and WebCT that sends periodic updates to WebCT via the Web-based IMS API. Updates could include students dropping and adding courses, the addition of new students, and the creation of courses that have a WebCT component.
4. Transfer of midterm and/or final grades from WebCT to the SIS via the Web-based IMS API. The transfer could occur when an instructor fills out a Web form indicating that grades are ready to be released to the registrar.

COMMAND LINE INTERFACE (EP_API.PL)

The IMS Best Practice and Implementation Guide describes a robust and easily implementable interface which involves the creation and passing of a complete "snapshot" of the Person, Group, and Group Membership data from one system to another. For example, at the beginning of a school year, an institution could export a snapshot of all student and course information from their SIS for the term. The "snapshot" could then be transferred to the WebCT server and imported.

The command line interface provides an effective way of importing "snapshots" into WebCT.

The command line interface also provides an effective way of exporting and configuring WebCT data without the development effort required for a Web-based implementation.

Note: *The IMS Best Practice and Implementation Guide* is available from http://www.imsproject.org/enterprise/

SYNTAX

The command line interface has the following general syntax:

```
ep_api.pl <ACTION> <OPTION> <FILENAME | COURSEID | GLOBAL ID>
      [--ims_id=<ID>] [--ims_source=<SOURCE>] [--ims_target=<TARGET>]
      [--datasource=<DATASOURCE>] [--studentlist=<STUDENTLIST>]
      [--sct_mode=<SCTMODE>] [--inline=<INLINEMODE>]
```

where:

ACTION	OPTION
import	restrict unrestrict
export	snapshot person_record group_record group_final_grades group_midterm_grades
configure	set_group_ims_info import_group_ims_info set_person_ims_info import_group_ims_info get_person_ims_info

FUNCTIONS

IMPORT

The syntax for an import is:

```
ep_api.pl import <OPTION> FILENAME [--inline=<INLINEMODE>]
      [--sct_mode=<SCTMODE>]
```

where:

Argument	Input	Description
OPTION	restrict	With restrict mode on, the sourcedid.source and sourcedid.ID supplied in the XML file are checked against the IMS sourcedid elements for similar objects to ensure they exist in the WebCT database. Objects can be updated or deleted only if the sourcedid.source element and sourcedid.ID elements match.
	unrestrict	No checking of the sourcedid.source element is performed; only the sourcedid.ID is checked.
FILENAME	filename	File to be imported into WebCT.
--inline	ON(default)	Objects are processed in the order they appear in the XML file.
	OFF	Group and Person objects are processed before Membership objects.
--sct_mode	ON (default)	Scenario 1: The following data elements are examined under GROUPTYPE: <TYPEVALUE level="2">Term</TYPEVALUE> <TYPEVALUE level="3">Course</TYPEVALUE> <TYPEVALUE level="4">Section</TYPEVALUE> A group object is added to the global database if the Term value appears but not the Course or Section values. Scenario 2: If the following data element is present under EXTENSION, the group object is added to the global database. <DELIVERY>WEBCT</DELIVERY> **Note**: If Scenario 2 dictates that a group object should be added to the global database, it will be added, regardless of the result of Scenario 1.
	OFF	No checking of TYPEVALUE or DELIVERY is performed.

Example

Import the XML file "course.xml" in restrict mode.

```
ep_api.pl import restrict courses.xml --sct_mode=OFF
```

EXPORT

This argument exports data from WebCT's global database. The syntax for an export is:

```
ep_api.pl export <OPTION> <FILENAME> [--datasource=<DATASOURCE>]
      [--ims_target=<TARGET>] [--type=<TYPE>] [--ims_id=<ID>]
      [--studentlist=<STUDENTLIST>]
```

where:

Argument	Input	Description
OPTION	snapshot	Create an XML file of all person, group, and membership objects.
	person_record	Create an XML file containing basic information for a single student record.
	group_record	Create an XML file containing a list of users with midterm and final grade information for a given course.
	group_final_grades	Create an XML file containing a list of users with final grade information for a given course.
	group_midterm_grades	Create an XML file containing a list of users with midterm grade information for a given course.
FILENAME	Any valid filename	Filename to be used for the XML file.
--datasource	Any alphanumeric string up to 256 characters. Enclose strings containing spaces in quotation marks.	Sets the DATASOURCE element within the PROPERTIES element. Defaults to "WebCT" if none is specified.
--ims_target	Any alphanumeric string up to 256 characters. Enclose strings containing spaces in quotation marks	Sets the TARGET element within the PROPERTIES element.
--type	Any alphanumeric string up to 256 characters. Enclose strings containing spaces in quotation marks.	Sets the TYPE element within the PROPERTIES object.
--ims_id	Any person or group object ID from the SOURCEDID data element	When exporting with the person_record, group_record, group_final_grades, or group_midterm_grades options, use this optional field to specify the person or group object that you want to export.
--studentlist	Any valid filename	When exporting using group_record, group_final_grades, or group_midterm_grades options, this optional file allows you to export a subset of the data. The file must be in plain text format with one IMS ID per line.

Example 1

Export a snapshot of the WebCT global database to the file dbsnap.xml with the datasource set to "WebCT - Faber College" and the target set to "BigSIS":

```
ep_api.pl export snapshot dbshot.xml --datasource="WebCT - Faber College"
      --ims_target="BigSIS"
```

Example 2

Export a person_record to the file person.xml for the user with the IMS ID "612":

```
ep_api.pl export person_record person.xml --ims_id=612
```

Example 3

Export a group_record to the file group.xml for a subset of students whose IMS IDs are stored in the file `students.txt`.

Note: The file containing the IMS IDs must be in plain text format, with one IMS ID per line.

```
ep_api.pl export group_record group.xml --studentlist=students.txt
    --ims_id=<ID>
```

CONFIGURE

Configure is used to set the IMS ID for group objects, and person objects. The syntax for a configure action is:

```
ep_api.pl configure <OPTION> <FILENAME | COURSEID | WEBCTID >
      [--ims_id=<ID>] [--ims_source=<SOURCE>]
```

where:

Argument	Input	Description
OPTION	set_group_ims_info	Sets the IMS ID for a group object (course)
	import_group_ims_info	Sets the IMS ID for multiple group objects from a file.
	set_person_ims_info	Sets the IMS ID for a person object (user).
	import_person_ims_info	Sets the IMS ID for multiple person objects from a file.
	get_person_ims_info	Finds a user's IMS ID and IMS source with the WebCT ID.
FILENAME	Any valid filename	A filename must be supplied for import_group_ims_info or import_person_ims_info. The file must be plain text, in the format: <webct_id>,<ims_id>,<ims_source_new>
COURSEID	An existing Course ID	A WebCT Course ID is required when using the set_group_ims_info option.
WEBCTID	An existing WebCT ID	A WebCT ID is required when using the set_person_ims_info option and the get_person_ims_info option.

--ims_id	Any valid IMS ID	For the set_group_ims_info and set_person_ims_info options, this argument allows you to set the Group→SourcedID→ID or Person→SourcedID→ID, respectively
--ims_source	Any valid IMS source	For the set_group_ims_info and set_person_ims_info options, this argument allows you to specify the desired Group→SourcedID→Source or Person→SourcedID→ID, respectively.

Example 1

Set the SourcedID→ID "0390-ENGL-101-2345" for a group object with the course ID "ENGL101-2345":

```
ep_api.pl configure set_group_ims_info ENGL101-2345
      --ims_id=0390-ENGL-101-2345
```

Example 2

Set the SourcedID→ID's for three person objects contained in the file `person_ims.txt`.

Note: When viewed with a text editor, the `person_ims.txt` file looks like the following:

```
bluto,blutarsky123,InstitutionSIS
pinto,kroger34,InstitutionSIS
flounder,dorfman53,InstitutionSIS

ep_api.pl configure import_person_ims_info person_ims.txt
```

WEB-BASED INTERFACE (SERVE_EP_API.PL)

The *IMS Best Practices and Implementation Guide* defines an event driven interface as an effective method of communicating periodic updates. The event-driven interface, in which events trigger the transmission of IMS data objects to the target system (e.g. WebCT), can be implemented effectively through the Web-based interface. For example, a student adding a course can trigger an event on an SIS which then sends an IMS data object to WebCT, updating the WebCT database.

Note: *The IMS Best Practices and Implementation Guide* is available from http://www.imsproject.org/enterprise

Implementing the Web-based interface involves two steps.

1. Setting the API shared secret value
2. Developing a program to generate an HTTP request

Step 1 can be accomplished by a WebCT administrator who has basic knowledge of the WebCT file system. Step 2 requires an experienced Web developer.

1. SETTING THE API SHARED SECRET VALUE

The shared secret value is a key component of allowing external servers to automatically sign on users to WebCT. The shared secret is used to create a Message Authentication Code (MAC) from the submitted data.

When WebCT receives a request, it decodes the shared secret from MAC using the submitted data. If the decoded shared secret is the same as the one stored locally, the request is considered authentic and is processed. You can set the shared secret by performing the following steps:

1. Using a text editor, open the file `<webct_install_directory>/webct/webct/generic/api/api_secret`
2. Change the first line of the file to your desired secret. (for security reasons, the default value "SECRET" does not work). You should note the following about the shared secret value.
   - It cannot exceed 256 characters.
   - It cannot contain tab, or other control characters.
   - It should not contain end-of-line characters. **Note**: By default, the UNIX text editor vi and pico automatically add end-of-line characters. Check the file size to ensure that the number of characters equals the number of bytes.
   - It is case-sensitive
3. Save the file.

Because the shared secret value has such a critical role, choose it carefully.

Tips for Shared Secrets

- Make your shared secret value difficult to guess by making it lengthy and by including a combination of numbers and upper and lower case characters.
- Change your shared secret value at regular intervals.
- On remote systems, place shared secret values in secure directories.

2. Developing a Program to Generate an HTTP Request

Developing a program to generate an http request is the most substantive part of implementing the Web-based IMS API. The program must:

- Generate a Message Authentication Code (MAC)
- Generate a checksum for submitted XML extracts
- Assemble a properly formatted http request
- Process any data being returned

Creating Message Authentication Codes

Because the Web interfaces to the Standard API and Automatic signon reside in public directories, Message Authentication Codes (MACs) are required to ensure that only messages from trusted servers are processed.

WebCT provides three options to assist you in creating MACs:

1. A C function that you may integrate and compile into your C program
2. An executable file to which you make a system call from your program
3. Instructions for generating a MAC using a language of your choice

Option 1: Using the get_authentication C Function

The get_authentication function generates a MAC from an array of data and a shared secret value.

The source code necessary to use the C function is located in `<webct_install_directory>/webct/webct/generic/api/security/`

A test program, which contains a Makefile for UNIX based systems, is also provided.

The file `api_security.c` contains the `get_authentication` function.

get_authentication	Generates a MAC from an array of data and a shared secret value
Syntax	char* get_authentication (int i, char* data[], char* secret, char* encrypted_data)
Returns	32 byte alphanumeric MAC

Parameter	Description
`i`	The number of elements in the array data[].
`data`	Array of all values to be used in generating the MAC. The data should not be URL encoded.
`secret`	The shared secret value.
`encrypted_data`	The memory location of the MAC. It must be at least 32 bytes long.

OPTION 2: USING MESSAGE AUTHENTICATION CODE GENERATOR (GET_MD5 EXECUTABLE)

The Message Authentication Code (MAC) generator generates a MAC from a shared secret value and a string consisting of the IMS ID, a timestamp, and a destination URL.

Use the Message Authentication Code (MAC) generator (an executable called get_md5) if you are not working in the C language, or do not want to create a function to create the MAC. You can make a system call to get_md5 from your program and have the authentication string returned. The get_md5 executable has no dependencies on WebCT and can be copied to other servers as required. If you need a get_md5 executable for an operating system other than the one your WebCT server is running on, you can download several pre-compiled binaries for other operating systems from `http://download.webct.com`

get_md5	Generates a MAC from a shared secret value and a line of data
Syntax	get_md5 <shared_secret_filename> <data_to_encrypt>

Returns	32 byte alphanumeric MAC
Attribute	**Description**
`shared_secret_filename`	The filename where the shared secret value is stored.
`string_to_encrypt`	The string to be encrypted. The string should not be URL encoded.

OPTION 3: CREATE A MAC USING A LANGUAGE OF YOUR OWN CHOICE

If you want to create MACs within your code, (e.g. you are writing your code in Java and don't want to make a system call), you can create a MAC with the following procedure:

1. Calculate the total of the ASCII values of all the characters in the data.
2. Convert the total of the ASCII values into a string.
3. Append the shared secret value.
4. Encrypt the string into a 16-byte string using the MD5 algorithm.
5. Convert the 16-byte string into a 32-byte alphanumeric string to make it URL-friendly.

Note: The FILENAME field should not be passed as data.

GENERATING A CHECKSUM

To ensure the integrity of XML files being transferred, WebCT uses a checksum. The checksum is created by summing the ASCII values of each character in the file (including line feeds and other control characters). As with other data, the checksum is used in the generation of the MAC. However, unlike other data, the checksum is not passed in the request to the Web server.

For example, the string "a dog\n" (where \n represents a line feed) has ASCII values of 97, 16, 100, 111, 103, and 10. The checksum for this string is 437, calculated by summing the values.

If you are programming in Perl, you can use the ord() function to get the ASCII value for a single character, and then loop through the entire file. In C, you can accomplish the same task using the atoi() function.

ASSEMBLING THE HTTP REQUEST

Your choice of language will determine the method that you use to assemble your http request. In general, you have two basic options:

- Socket programming with the Web server
- Using a library which simulates a user agent

In Perl, you have the option of communicating directly with the Web server using the IO::Socket module included with most basic distributions, or installing and using a module such as LWP which simulates a user agent (e.g. a Web browser). Similar modules are available for most popular languages such as C or Java. Although no language is recommended over others, the examples in this document use Perl and the IO::Socket module to communicate with the Web server.

The serve_ep_api.pl CGI will accept both GET and POST requests. However, anytime that you need to submit a file to the server, a POST request will be necessary in since GET requests are limited in length. Actions that require you to use POST are:

- Import actions
- Export actions that utilize the STUDENTLIST option
- The import_group_ims_info and import_person_ims_info actions.

The following example (written in Perl) imports an XML extract into WebCT. It generates a MAC for the data, generates a checksum for the XML extract, POSTs an http request to the serve_ep_api.pl CGI via sockets, and prints out the response from the Web server.

```
#!/usr/bin/perl
use strict;
use IO::Socket;

my $remote_host = "webct.institution.com";
my $remote_port = "8900";
my $SECRET_FILE = "api_secret";

my %params;

# Normally, you wouldn't hard code the following values, but
# since this program is only trying to show a example of how
# to communicate with the API, we can get away with it.
$params{"FILENAME"} = "extract.xml";
$params{"ACTION"} = "import";
$params{"OPTION"} = "restrict";
$params{"SCTMODE"} = "OFF";          # ON by default

# Read in the XML extract and assign it to a variable
my $filename_data = &read_file($params{"FILENAME"});

# Calculate a timestamp
$params{"TIMESTAMP"} = time();
# Print out the TIMESTAMP for debugging purposes
print "TIMESTAMP: " . $params{"TIMESTAMP"} . "\n";

# Generate a checksum from the calculate_checksum subroutine
$params{"CHECKSUM"} = &calculate_checksum($filename_data);
# Print out the checksum for debugging purposes
print "Checksum: " . $params{"CHECKSUM"} . "\n";

# Concatenate the data into a single string so that we can
# create the MAC
my $data_string = $params{"ACTION"};
$data_string .= $params{"OPTION"};
$data_string .= $params{"TIMESTAMP"};
$data_string .= $params{"CHECKSUM"};
$data_string .= $params{"SCTMODE"};

# Make a system call to the program that generates MACs
$params{"AUTH"} = `./get_md5 $SECRET_FILE $data_string`;
# Print out the MAC for debugging purposes
print "Auth: ".$params{"AUTH"}."\n";
```

```
# To make multipart/form-data requests, we need to use a boundary
# that won't interfere with our data; you can choose anything you like.
my $content_boundary = "WebCT_Enterprise_API_Boundary";

# This script is going to post all data in the body of the message
# The $file_content variable will hold the body.
my $file_content;

$file_content .= "--".$content_boundary."\r\n";
$file_content .= "Content-Disposition: form-data; name=\"ACTION\"\r\n\r\n";
$file_content .= "$params{'ACTION'}\r\n";
$file_content .= "--".$content_boundary."\r\n";
$file_content .= "Content-Disposition: form-data; name=\"OPTION\"\r\n\r\n";
$file_content .= "$params{'OPTION'}\r\n";
$file_content .= "--".$content_boundary."\r\n";
$file_content .= "Content-Disposition: form-data; name=\"TIMESTAMP\"\r\n\r\n";
$file_content .= "$params{'TIMESTAMP'}\r\n";
$file_content .= "--".$content_boundary."\r\n";
$file_content .= "Content-Disposition: form-data; name=\"AUTH\"\r\n\r\n";
$file_content .= "$params{'AUTH'}\r\n";
$file_content .= "--".$content_boundary."\r\n";
$file_content .= "Content-Disposition: form-data; name=\"SCTMODE\"\r\n\r\n";
$file_content .= "$params{'SCTMODE'}\r\n";
$file_content .= "--".$content_boundary."\r\n";
$file_content .= "Content-Disposition: form-data; name=\"FILENAME\";";
$file_content .= " filename=\"$params{'FILENAME'}\"\r\n";
$file_content .= "Content-Type: text/xml\r\n\r\n";
$file_content .= "$filename_data\r\n";
$file_content .= "--".$content_boundary."--\r\n";

# Open the socket connection to the Web server
my $socket = IO::Socket::INET->new(PeerAddr => "$remote_host:$remote_port",
                    PeerPort => $remote_port,
                    Proto    => "tcp",
                    Type     => SOCK_STREAM);

if (!$socket)
{
print "Could not open connection to $remote_host:$remote_port";
return 0;
}

# Determine the size of the message body for the
# http Content-length header
my $file_size = length($file_content);

# Send the request to the Web server
print $socket "POST /webct/ims/serve_ep_api.pl HTTP/1.0\n";
print $socket "Content-length: $file_size\n";
print $socket "Content-type: multipart/form-data;
      boundary=$content_boundary\n\n";
print $socket $file_content;


# get results
my @results;
```

```
@results = <$socket>;

# Print the results to standard output. A real program would
# want to do some processing of the results to check for success
# or failure.
print @results;

# Close the socket connection
close($socket);

#####################################################################
# read_file
# Reads in a file, returns the file's contents as a variable
#####################################################################
sub read_file
{
    my $filename = shift(@_);
    my $file_content = undef;

    if(open(FH,$filename)) {
      while (my $line = <FH>)
      {
          $file_content .= $line;
      }
      close(FH);
    }

    return $file_content;
}

#####################################################################
# calculate_checksum
#
#####################################################################
sub calculate_checksum
{
    my ($data) = @_;
    my ($checksum) = 0;

    # Sum up the ASCII values of all the characters in $data
    while ($data)
    {
      $checksum = $checksum + ord($data);
      $data = substr($data, 1);
    }

    return $checksum;
}
```

XML File Format Guidelines

The IMS Enterprise API is based on the exchange of XML files between IMS Enterprise-compliant systems. Each XML file contains one or more data objects, which each represent an operation that should occur (e.g. add a user to the database, create a new course instance). All IMS Enterprise documents have the following general structure:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE ENTERPRISE SYSTEM "IMS-EP01.dtd" >

<ENTERPRISE>

      <OBJECT1>
      </OBJECT1>

      <OBJECT2>
      </OBJECT2>

      <OBJECTn>
      </OBJECTn>

</ENTERPRISE>
```

This example demonstrates a "shell" file that does not perform any actions. In this example, the three OBJECT tags represent placeholders that would be filled with properties, group, person, or membership data objects. You can place as many data objects as you want within the file.

IMS Objects and WebCT Relationships

All objects in an IMS-compliant XML file are based on the IMS Enterprise Information Model (available at http://www.imsproject.org/enterprise). Because the IMS Information Model is very broad to cover a wide range of needs, WebCT only uses a subset of information from the model. The following sections outline the relationship between the IMS data objects and WebCT data.

Properties Object

The properties object contains some general packaging and control data for use by the target system (WebCT). The following is a fragment from an XML file showing a properties object:

```
<PROPERTIES>
  <DATASOURCE>Faber College SIS</DATASOURCE>
  <DATETIME>2000-12-21</DATETIME>
</PROPERTIES>
```

The following table describes data elements relevant to WebCT:

Data Element	Required	Description
Datasource	Yes	The Properties→Datasource element is the identifier of the system that generated the XML file.
Datetime	Yes	Although not used by WebCT, a Datetime element with date and time in ISO8601 standard format is required for IMS-compliance.

GROUP OBJECT

The following is a code fragment from an XML file. It shows a group object:

```
<GROUP recstatus="1">
      <SOURCEDID>
        <SOURCE>Faber College SIS</SOURCE>
        <ID>0390COMPSCI697CSec1-1164</ID>
      </SOURCEDID>
      <DESCRIPTION>
        <SHORT>Security In Computing</SHORT>
      </DESCRIPTION>
      <EXTENSION>
        <TEMPLATE>Blank</TEMPLATE>
      </EXTENSION>
</GROUP>
```

The following table describes data elements relevant to WebCT:

Data Element	Required	Description
RecStatus	No	This describes the type of action to be performed on an object. Numbers are used for language independence: 1 = Add, 2 = Update, and 3 = Delete. If no RecStatus is supplied, the API will default to 1 (Add) if the record does not already exist, or 2 (Update) if the record does exist.
Source	Yes	The Group→SourcedID→Source element is used when importing in restrict mode. This value is compared against the IMS sourcedid.source element for the item on the WebCT server to determine if an object will be processed.
ID	Yes	The Group→SourcedID→ID element is used as the WebCT Course ID.
Short	Yes	The Group→Description→Short element is used as the Course Description within WebCT.
Template	No	The Group→Extension→Template element allows you to specify a course template or course ID as the basis for another course. It is processed in the following order: 1. If no Template element is specified, use the SCT template. 2. If this tag has the text "Blank," a blank template is used. 3. If a valid Course ID is supplied, the course is created based on it. 4. If a valid course template ID is supplied, it is used. Valid course template IDs are "blank", "photo_basic"(basic), "photo"(intermediate), and "photo_comprehensive"(advanced) 5. If an invalid value is supplied, use the default template.

SPECIFIC OBJECT TYPES

One Object type, the term object (school term), is described in the section that follows.

ABOUT TERMS

The IMS API provides a means to add, update, and delete terms, as well as to add a course to a term and to add a course to a category. Functionality related to terms is also provided through the Administrative Interface. Note that terms initially created through the IMS API cannot be deleted through the Administrative Interface; deletion must be performed using the IMS API.

TERM OBJECT

The following is an XML code fragment showing two term objects. To add a term, prepare an XML file using these guidelines and add to the database using the 'import' command.

```
<GROUP>
   <SOURCEDID>
      <SOURCE>Faber College SIS</SOURCE>
      <ID>2002-Summer</ID>
   </SOURCEDID>
<GROUPTYPE>
    <TYPEVALUE level="1">Instruction</TYPEVALUE>
    <TYPEVALUE level="2">Term</TYPEVALUE>
</GROUPTYPE>
<DESCRIPTION>
   <SHORT>2002 Term 2</SHORT>
   <LONG>Summer 2002</LONG>
</DESCRIPTION>
</GROUP>
<GROUP>
   <SOURCEDID>
      <SOURCE>Faber College SIS</SOURCE>
      <ID>2002-Fall</ID>
 </SOURCEDID>
<GROUPTYPE>
   <TYPEVALUE level="1">Instruction</TYPEVALUE>
   <TYPEVALUE level="2">Term</TYPEVALUE>
</GROUPTYPE>
<DESCRIPTION>
   <SHORT>2002 Term 3</SHORT>
   <LONG>Fall 2002</LONG>
</DESCRIPTION>
</GROUP>
```

The following table describes data elements within the code fragment that are relevant to WebCT:

Data Element	Required	Description
RECSTATUS	No	This describes the type of action to be performed on an object. Numbers are used for language independence: 1 = Add, 2 = Update, and 3 = Delete. If no RecStatus is supplied, the API will default to 1 (Add) if the record does not already exist, or 2 (Update) if the record does exist.

SOURCE	Yes	The Group →SourcedID→Source element is used when importing in restrict mode. This value is compared against the IMS sourceid.source element for the item on the WebCT server to determine if an object will be processed.
ID		The Group → SourcedID→ID element is used as the WebCT ID and is stored internally. The WebCT ID can be overridden by the UserID element.
TYPEVALUE	Yes	The Group→Typevalue element is required to indicate that the group object is a term. Each TYPEVALUE element requires a level to be defined. Level="1" specifies this is an instructional group object. Level="2" further specifies the instructional group object (in the example given) as a term object.
SHORT	Yes	Sorting key for term element. For example, Spring, Summer, Fall, Winter terms could be assigned values 1, 2, 3, and 4 respectively to ensure the terms are displayed in the correct order in the WebCT interface.
LONG	No, but recommended	Term title that will appear in WebCT

ASSIGNING A COURSE TO A CATEGORY

The following is an XML code fragment showing the assignment of a course within a category:

```
<?xml version="1.0" encoding="UTF-8" ?>
  <GROUP>
     <SOURCEDID>
        <SOURCE>Faber College SIS</SOURCE>
        <ID>0390COMPSCI697CSec1-1164</ID>
     </SOURCEDID>
     <DESCRIPTION>
        <SHORT>CS-697</SHORT>
        <LONG>Security In Computing</LONG>
     </DESCRIPTION>
    <ORG>
      <ORGNAME>Faber College</ORGNAME>
      <ORGUNIT>Physics</ORGUNIT>
    </ORG>
```

The following table describes data elements within the code fragment that are relevant to WebCT:

Data Element	Required	Description
RECSTATUS	No	This describes the type of action to be performed on an object. Numbers are used for language independence: 1 = Add, 2 = Update, and 3 = Delete. If no RecStatus is supplied, the API will default to 1 (Add) if the record does not already exist, or 2 (Update) if the record does exist.
SOURCE		The Group→SourcedID→Source element is used when importing in restrict mode. This value is compared against the IMS sourceid.source element for the item on the WebCT server to determine if an object will be processed.

ID		The Group → SourcedID→ID element is used as the WebCT ID and is stored internally. The WebCT ID can be overridden by the UserID element.
SHORT	Yes	This is the course title.
LONG	No, but recommended	This is the course description.
ORG	No	Needed when assigning a course to a category.
ORGNAME	Yes	Institution name (this information is not actually used by WebCT)
ORGUNIT	Yes	The course will be assigned to the Category with this name. If your institution is organized by Departments, the ORGUNIT may be Physics, or Psychology.

ASSIGNING A COURSE TO A TERM

The following is an XML code fragment showing the assignment of a course to a term.

```
<GROUP>
  <SOURCEDID>
      <SOURCE>Faber College SIS</SOURCE>
      <ID>0390COMPSCI697CSec1-1164</ID>
  </SOURCEDID>
  <DESCRIPTION>
      <SHORT>CS-697</SHORT>
      <LONG>Security In Computing</LONG>
  </DESCRIPTION>
  <RELATIONSHIP myrelationship="1">
    <SOURCEDID>
      <SOURCE>Faber College SIS</SOURCE>
      <ID>2002-Summer</ID>
    </SOURCEDID>
  </RELATIONSHIP>
```

The following table describes data elements within the code fragment that are relevant to WebCT:

Data Element	Required	Description
RECSTATUS	No	This describes the type of action to be performed on an object. Numbers are used for language independence: 1 = Add, 2 = Update, and 3 = Delete. If no RecStatus is supplied, the API will default to 1 (Add) if the record does not already exist, or 2 (Update) if the record does exist.
SOURCE		The Group →SourcedID→Source element is used when importing in restrict mode. This value is compared against the IMS sourceid.source element for the item on the WebCT server to determine if an object will be processed.
ID		The Group→ SourcedID→ID element is used to generate

		the Course ID and is stored internally.
SHORT	Yes	This is the course title.
LONG	No, but recommended	This is the course description.
RELATIONSHIP	Yes	myrelationship="1" indicates that the course specified should be related to the term indicated.
SOURCE		The Group →Relationship→Source element is used when importing in restrict mode. This value is compared against the IMS sourceid.source element for the term to which the course is being assigned.
ID		The Group→ Relationship→SourcedID→ID element refers to the Group→ SourcedID→ID of the term this course is being assigned to.

PERSON OBJECT

The following is an XML code fragment showing a person object:

```
<PERSON recstatus="1">
      <SOURCEDID>
        <SOURCE>Faber College SIS</SOURCE>
        <ID>39450210223</ID>
      </SOURCEDID>
      <USERID>HooverR</USERID>
      <NAME>
      <FN>Robert Hoover</FN>
        <N>
              <FAMILY>Hoover</FAMILY>
              <GIVEN>Robert</GIVEN>
        </N>
      </NAME>
      <EXTENSION>
        <WEBCREDENTIAL>ToughPassword</WEBCREDENTIAL>
      </EXTENSION>
</PERSON>
```

The following table describes data elements relevant to WebCT:

Data Element	Required	Description
RecStatus	No	This describes the type of action to be performed on an object. Numbers are used for language independence. 1 = Add, 2 = Update, and 3 = Delete. If no RecStatus is supplied, the API will default to 1 (Add) if the record does not already exist, or 2 (Update) if the record does exist.
Source	Yes	The Person→SourcedID→Source element is used when importing in restrict mode. This value is compared against the IMS sourceid.source element for the item on the WebCT server to determine if an object will be processed.

ID	Yes	The Person→ SourcedID→ID element is used as the WebCT ID and is stored internally. The WebCT ID can be overridden by the UserID element.
UserID	No	The option Person→ UserID element can specify a WebCT ID to be used.
FN	Yes	The Person→ Name→FN (Formatted Name) element is required by the IMS Specification and is stored internally. It is only used for compliance.
Family	No	The Person→Name→N→Family element maps to the Last Name field within the WebCT global database.
Given	No	The Person→ Name→N→Given element maps to the First Name field within the WebCT global database.
WebCredential	No	The Person→ Extension→WebCredential data element maps to the Password field within WebCT. If this element is not supplied, the password defaults to the WebCT ID given to the user.

MEMBERSHIP OBJECT

The following is an XML code fragment showing a membership object with two member elements underneath it (one student and one designer).

```
<MEMBERSHIP>
  <SOURCEDID>
        <SOURCE>Faber College SIS</SOURCE>
        <ID>0390COMPSCI697CSec1-1164</ID>
  </SOURCEDID>
  <MEMBER>
        <SOURCEDID>
              <SOURCE>Faber College SIS</SOURCE>
              <ID>39450210223</ID>
        </SOURCEDID>
        <IDTYPE idtype="1"/>
        <ROLE recstatus="1" roletype="01">
              <USERID>39450210223</USERID>
              <STATUS>1</STATUS>
              <FINALRESULT>
                    <RESULT>B</RESULT>
              </FINALRESULT>
              <EXTENSION>
                    <MIDTERMRESULT>
                          <RESULT>A</RESULT>
                    <MIDTERMRESULT>
              </EXTENSION>
        </ROLE>
  </MEMBER>
  <MEMBER>
        <SOURCEDID>
```

```
            <SOURCE>Faber College SIS</SOURCE>
            <ID>012345678910</ID>
        </SOURCEDID>
        <IDTYPE idtype="1"/>
        <ROLE recstatus="1" roletype="02">
            <USERID>12345678910</USERID>
            <SUBROLE>Primary</SUBROLE>
            <STATUS>1</STATUS>
        </ROLE>
    </MEMBER>
</MEMBERSHIP>
```

The following table describes data elements relevant to WebCT:

Data Element	Required	Description
Membership→SourcedID→ID	Yes	This Membership→SourcedID→ID should match an existing Group→SourcedID→ID element. Users will be manipulated in the WebCT course instance that this matches.
Member	No	The Membership→Member element can be repeated multiple times to associate multiple users with the group specified in Membership→SourcedID→ID.
Membership→Member →SourcedID→Source	Yes	The Membership→Member→SourcedID→Source element is used when importing in restrict mode. This value is compared against the IMS sourceid.source element for the item on the WebCT server to determine if an object will be processed.
Membership→Member →SourcedID→ID	Yes	This ID should match an existing Person→SourcedID→ID. This is the person object being added, updated, or deleted within the course.
IDType	Yes	The Membership→Member→IDType element is required according to the IMS specification and indicates if the member is a person (indicated by a "1") or a group object (indicated by a "2"). WebCT only supports person objects.
RecStatus	No	This describes the type of action to be performed on an object. Numbers are used for language independence. 1 = Add, 2 = Update, and 3 = Delete. If no RecStatus is supplied, the API will default to 1 (Add) if the record does not already exist, or 2 (Update) if the record does exist.
RoleType	Yes	The Membership→Member→Role→RoleType determines what type of user this person object should be. 01 = Learner/Student, 02 = Instructor. Other numbers listed in the IMS Specification are not currently supported.
UserID	Conditional	The Membership→Member→Role→UserID is a required element if the associated person object has a UserID associated with it. The UserIDs in both objects must match or an error will be returned.
SubRole	No	For instructors/designers, a Membership→Member→Role→Subrole can be specified as "Primary" or "Subordinate" that map in

Data Element	Required	Description
		WebCT to primary and secondary designers. The IMS specifications also suggest that "Teaching Assistant" may be used as a SubRole. However, WebCT's IMS API does not yet support Teaching Assistants.
Status	Yes	This Membership→Member→Role→Status element affects students (but must be included for all users for IMS-compliance). "1" indicates that a user is active, and "0" indicates inactive. Students with inactive status will be denied access and their records within Manage Students will appear in gray.
Membership→Member→Role →FinalResult→Result	No	The Membership→Member→Role→FinalResult→Result element maps to the Final Grade column within Manage Students
Membership→Member →Extension→MidtermResult →Result	No	This element maps to the Midterm column within Manage Students.

COMPLETE SPECIFICATIONS

A complete discussion of the IMS Enterprise Information Model is beyond the scope of this document. You should refer to the IMS Enterprise Information Model, Version 1.01, available at http://www.imsproject.org/enterprise.

OTHER XML CONSIDERATIONS

In addition to following the IMS specification for XML, WebCT requires that documents are well formed and follow XML convention. Any error that causes WebCT's XML parser to fail will result in the entire API action failing with errors generated to standard error (on screen for command line operations, and to the Apache error logs for Web-based requests). Applications that generate XML markup should ensure that they are following the XML 1.0 specification (available from the W3C at http://www.w3.org/TR/REC-xml).

SYNTAX

The general syntax for a Web-based request to the IMS API is as follows:

```
<GET | POST>  /webct/ims/serve_ep_api.pl?ACTION=<action>&OPTION=<option>
      &TIMESTAMP=<unix_epoch_time>&AUTH=<32_byte_mac>
      [&INLINEMODE=<inlinemode>][&COURSE=<course>]
      [&DATASOURCE=<datasource>][&TARGET=<target>][&ID=<id>]
      [&SOURCE=<source>][&SCTMODE=<sctmode>] HTTP/1.0

[--Boundary_Value_Of_Your_Choosing]
[Content-Disposition: form-data; name="FILENAME"; filename="<filename>"
Content-Type: text/xml | Content-Type: text/plain

<file_content>]
[--Boundary_Value_Of_Your_Choosing]
[Content-Disposition: form-data; name="STUDENTLIST";
      filename="<studentlistfilename>"
Content-Type: text/plain

<studentlist_file_content>]
```

where:

ACTION	Description
import	Imports an XML extract into the WebCT global database.
export	Exports an XML extract from the WebCT global database.
configure	Configures the IMS ID for a person or group object

Notes:

- Although either GET or POST methods may be used, any request that requires the transfer of a file requires you to use POST
- Requests using the POST method may pass the key/value pairs to the Web server using the application/x-www-form-urlencoded content type or via the multipart/form-data content type within the body of the message. If you are uploading a file, the multipart/form-data content-type is required.
- Syntax examples represent http requests directly to the Web server. If you are using a programming module to create your requests (such as LWP in Perl), many details of the request may be transparent to you.
- Because POSTing in multipart/form-data is extremely verbose, syntax examples have been consolidated so that only files to be uploaded appear in full syntax. Other key/value pairs are presented in the query string. In practice, your requests must send all data in the message body when POSTing.

FUNCTIONS

IMPORT

This action imports data to the WebCT global database. The syntax for an import is:

```
POST  /webct/ims/serve_ep_api.pl?ACTION=import&OPTION=<option>
      &TIMESTAMP=<unix_epoch_time>&AUTH=<32_byte_mac>
      &CHECKSUM=<checksum>[&INLINEMODE=<inlinemode>[&SCTMODE=<sctmode>] HTTP/1.0

--Boundary_Value_Of_Your_Choosing
Content-Disposition: form-data; name="FILENAME" filename="<filename>"
Content-Type: text/xml

<file_content>
--Boundary_Value_Of_Your_Choosing--
```

where:

Key/ Parameter	Value	Description
OPTION	restrict	With restrict mode on, the sourcedid.source and sourcedid.ID supplied in the XML file are checked against the IMS sourcedid elements for similar objects to ensure they exist in the WebCT database. Objects can be updated or deleted only if the sourcedid.source element and sourcedid.ID elements match.
	unrestrict	No checking of the sourcedid.source element is performed; only the sourcedid.ID is checked.
TIMESTAMP	UNIX epoch timestamp	Time stamp in UNIX epoch format (seconds since midnight GMT, Jan 1, 1970)
AUTH	A valid MAC	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
INLINE	ON	Objects are processed in the order they appear in the XML file.
	OFF (default)	Group and Person objects are processed before Membership objects.

Key/ Parameter	Value	Description
SCTMODE	ON (default)	Scenario 1: The following data elements are examined under GROUPTYPE: <TYPEVALUE level="2">Term</TYPEVALUE> <TYPEVALUE level="3">Course</TYPEVALUE> <TYPEVALUE level="4">Section</TYPEVALUE> A group object is added to the global database if the Term value appears but not the Course or Section values. Scenario 2: If the following data element is present under EXTENSION, the group object is added to the global database: <DELIVERY>WEBCT</DELIVERY> **Note**: If Scenario 2 dictates that a group object should be added to the global database, it will be, regardless of the result of Scenario 1.
	OFF	No checking of TYPEVALUE or DELIVERY is performed.
<filename>	A valid filename	Filename to be imported into WebCT.
<file_content>	Contents of XML extract.	The contents of an IMS-compliant XML file.

Example

This example imports an XML file that includes a single person object.

```
POST /webct/ims/serve_ep_api.pl HTTP/1.0
Content-length: 1253
Content-type: multipart/form-data; boundary=WebCT_Enterprise_API_Boundary

--WebCT_Enterprise_API_Boundary
Content-Disposition: form-data; name="ACTION"

import
--WebCT_Enterprise_API_Boundary
Content-Disposition: form-data; name="OPTION"

restrict
--WebCT_Enterprise_API_Boundary
Content-Disposition: form-data; name="TIMESTAMP"

984693507
--WebCT_Enterprise_API_Boundary
Content-Disposition: form-data; name="AUTH"

68056FBF19C2C5FE3B7AB63A06B9A009
--WebCT_Enterprise_API_Boundary
Content-Disposition: form-data; name="SCTMODE"
```

```
OFF
--WebCT_Enterprise_API_Boundary
Content-Disposition: form-data; name="FILENAME"; filename="event.xml"
Content-Type: text/xml

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE ENTERPRISE SYSTEM "IMS-EP01.dtd" >

<ENTERPRISE>
      <PROPERTIES lang="EN">
        <DATASOURCE>Faber College SIS</DATASOURCE>
        <DATETIME>2001-03-04</DATETIME>
      </PROPERTIES>

      <PERSON>
        <SOURCEDID>
              <SOURCE>Faber College SIS</SOURCE>
              <ID>Hoover26</ID>
        </SOURCEDID>
        <USERID>rhoover</USERID>
        <NAME>
              <FN>Robert Hoover</FN>
              <N>
                    <FAMILY>Hoover</FAMILY>
                    <GIVEN>Robert</GIVEN>
              </N>
        </NAME>
        <DATASOURCE>Faber College SIS</DATASOURCE>
        <EXTENSION>
              <WEBCREDENTIAL>ToughPassword</WEBCREDENTIAL>
        </EXTENSION>
      </PERSON>

</ENTERPRISE>

--WebCT_Enterprise_API_Boundary—
```

The Web server returns (not including http headers):

```
Success: Data successfully imported.
Success: Import complete.
```

EXPORT

This action exports data from the WebCT global database. The syntax for an export is:

```
<GET | POST>  /webct/ims/serve_ep_api.pl?ACTION=export&OPTION=<option>
      &TIMESTAMP=<unix_epoch_time>&AUTH=<32_byte_mac>
      [&DATASOURCE=<datasource>]
      [&TARGET=<target>][&TYPE=<type>][&ID=<id>]
      [&STUDENTLIST=<studentlist>]
      [&STUDENTLISTCHECKSUM=<studentlistchecksum>] HTTP/1.0

[--Boundary_Value_Of_Your_Choosing
Content-Disposition: form-data; name="STUDENTLIST";
      filename="<studentlist_filename>"
Content-Type: text/plain

<studentlist_file_content>
--Boundary_Value_Of_Your_Choosing--]

where:
```

Key/Parameter	Input	Description
OPTION	snapshot	Creates an XML file of all person, group and membership objects
	person_record	Creates an XML file containing basic information for a single student record
	group_record	Creates an XML file containing a list of users with midterm and final grade information for a given course
	group_final_grades	Creates an XML file containing a list of users with final grade information for a given course
	group_midterm_grades	Creates an XML file containing a list of users with midterm grade information for a given course
TIMESTAMP	UNIX epoch timestamp	Time stamp is in UNIX epoch format (seconds since midnight GMT, Jan 1, 1970)
AUTH	A valid MAC	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
DATASOURCE	Any alphanumeric string up to 256 characters. Strings containing spaces can be enclosed in quotation marks.	Sets the DATASOURCE element within the PROPERTIES element. Defaults to "WebCT" if none is specified.
TARGET	Any alphanumeric string up to 256 characters. Strings	Sets the TARGET element within the PROPERTIES object

Key/Parameter	Input	Description
	containing spaces can be enclosed in quotation marks.	
TYPE	Any alphanumeric string up to 256 characters. Strings containing spaces can be enclosed in quotation marks.	Sets the TYPE element within the PROPERTIES object
ID	Any person or group object SourcedID→ID.	When exporting with the person_record, group_record, group_final_grades, or group_midterm_grades options, you use this optional field to specify the person or group object that you want to export
<studentlist_filename>	Any valid filename	This optional element is provided when using the group_record, group_final_grades, or group_midterm_grades options when you wish to export a subset of data.
<studentlist_file_content>	Contents of file	This optional element is provided when using the group_record, group_final_grades, or group_midterm_grades options when you wish to export a subset of data. The file must be in plain text, with one IMS ID per line.

Example

The following request generates an XML file for a person object with the Person→SourcedID→ID of "Hoover26":

```
GET  /webct/ims/serve_ep_api.pl?ACTION=export&OPTION=person_record
      &TIMESTAMP=984694373&ID=Hoover26
      &AUTH=1F2E7CA6B6EBCE62D3FC089CA42E80FB HTTP/1.0
```

The server returns the following (not including http headers):

```
<?xml version="1.0" encoding="UTF-8"?>

<ENTERPRISE>
  <PROPERTIES>
    <DATASOURCE>WebCT</DATASOURCE>
    <DATETIME>2001-03-15T14:15:18-0800</DATETIME>
  </PROPERTIES>
  <PERSON>
    <SOURCEDID>
       <SOURCE>Faber College SIS</SOURCE>
       <ID>Hoover26</ID>
    </SOURCEDID>
    <USERID>rhoover</USERID>
```

```
    <NAME>
      <FN>Robert Hoover</FN>
      <N>
        <FAMILY>Hoover</FAMILY>
        <GIVEN>Robert</GIVEN>
      </N>
    </NAME>
    <DATASOURCE>WebCT</DATASOURCE>
  </PERSON>
</ENTERPRISE>
```

CONFIGURE

Configure sets the IMS ID for group objects and person objects. The syntax for configure is:

```
<GET | POST>  /webct/ims/serve_ep_api.pl?ACTION=configure&OPTION=<option>
      &TIMESTAMP=<unix_epoch_time>&AUTH=<32_byte_mac>
      [&COURSE=<course>][&GLOBALID=<WebCTID>][&ID=<id>][&SOURCE=<source>]

[--Boundary_Value_Of_Your_Choosing
Content-Disposition: form-data; name="FILENAME"; filename="<filename>"
Content-Type: text/plain

<file_content>
--Boundary_Value_Of_Your_Choosing--]
```

where:

Argument	Input	Description
OPTION	set_group_ims_info	Sets the IMS ID for a group object (course)
	import_group_ims_info	Sets the IMS ID for multiple group objects (courses) from a file
	set_person_ims_info	Sets the IMS ID for a person object (user)
	import_person_ims_info	Sets the IMS ID for multiple person objects (users) from a file
	get_person_ims_info	Finds the IMS ID and IMS source for a single user
TIMESTAMP	UNIX epoch timestamp	Time stamp in UNIX epoch format (seconds since midnight GMT, Jan 1, 1970)
AUTH	A valid MAC	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
COURSE	An existing Course ID	An existing WebCT Course ID is required when using the set_group_ims_info option.
GLOBALID	An existing WebCT ID	An existing WebCT ID is required when using the `set_person_ims_info` option and the `get_person_ims_info` option
ID	Any valid IMS ID	This argument sets the Group→SourcedID→ID or

Argument	Input	Description
		Person→SourcedID→ID for the set_group_ims_info and `set_person_ims_info` options, respectively.
SOURCE	Any valid IMS source	This argument sets the Group→SourcedID→Source or Person→SourcedID→Source for the `set_group_ims_info` and `set_person_ims_info` options, respectively.
<filename>	A valid filename	A filename must be supplied for `import_group_ims_info` or `import_person_ims_info`.
<file_content>	Contents of text file	The file must be plain text, in the format: <webct_id>,<ims_id>,<ims_source_new>.

Example

This example sets the Person→Sourced→ID to "Pepperidge23" and the Person→SourcedID→Source to "Faber College SIS" for the WebCT ID "mpepperidge":

```
GET  /webct/ims/serve_ep_api.pl?ACTION=configure
      &OPTION=set_person_ims_info&TIMESTAMP=984701570
      &GLOBALID=mpepperidge&ID=Pepperidge23
      &SOURCE=Faber%20College%20SIS
      &AUTH=95F5858984AB4D1571DC5BE9BD8E21DB HTTP/1.0
```

The Web server returns (not including http headers):

```
<RESPONSE responsetext="optional">SUCCESS</RESPONSE>
Success: IMS info updated for mpepperidge.
```

CHAPTER 5
STANDARD API

The Standard API gives administrators and developers access to the WebCT databases via command line or Web-based interfaces. The Standard API can be used to integrate external applications with WebCT. For example, it can be used for integrating WebCT with a Student Information System.

FUNCTIONALITY IN THE STANDARD API

The Standard API allows you to manipulate two separate databases within WebCT, the global database and the student database.

The global database contains the central listing of users for all users on the WebCT server. By default, the global database contains the WebCT ID, Password, First Name, Last Name, Courses, and Registered Courses fields. All users must have an entry in this database in order to access a course.

The student database is a term for a collection of databases specific to a course. Every WebCT course has its own student database that contains, by default, the User ID, Password, First Name, and Last Name fields. The information in the student database can be viewed most readily by looking at the designer interface of Manage Students.

Generally, a WebCT ID is linked to a User ID for each course that a user is enrolled in. Users can have different User IDs from their WebCT IDs, as well as different First Name and Last Name data in the student and global databases.

The functionality of the Standard API can be divided into the following basic categories:

Adding Users	• Adding a single user to the global database or student database • Adding multiple users to the global database or student database
Updating Users	• Updating a single user in the global database or student database • Updating multiple users in the global database or student database • Updating user types
Deleting Users	• Deleting a single user from the global database or student database • Deleting multiple users from the global database or student database
Finding WUUIs	• Finding the WebCT Unique Universal Identifier in the global database, either by WebCT ID or IMS ID **Important**: Since the release of WebCT 3.6, the use of the WUUI for Automatic Signon and the `find_wuui` operation are deprecated. With WebCT moving towards the use of the IMS specifications, which are becoming standards in the learning community, the IMS ID and IMS source are now preferred over the WUUI. Although the use of the WUUI is deprecated, the functionality remains.
Finding Users	• Finding a user in the global database or student database

Changing WebCT IDs	• Changing a single user's WebCT ID • Changing multiple users' WebCT IDs
Exporting *myWebCT* in XML format	• Exporting a user's *myWebCT* in XML format

IMPLEMENTING THE STANDARD API

COMMAND LINE INTERFACE (WEBCTDB)

The command line interface to the standard API provides a simple interface to the WebCT API. The executable file `webctdb`, is located in the directory `<install_dir>/webct/webct/generic/api/.`

SYNTAX

The general syntax for each of the Standard API operations is as follows:

Operation	Field Names
add	<db> <course> <fieldsData_pair_list> <separator> [encrypted]
delete	<db> <course> <WebCT_ID | user_id>
changeid	<db> <course> <fieldsData_pair_list> <separator>
update	<db> <course> <fieldsData_pair_list> <separator> [encrypted]
find	<db> <course> <WebCT_ID | user_id> <separator> [user_type]
find_wuui	<db> <course> <WebCT_ID>
find_ims_id_wuui	<db> <course> <IMS_ID>
fileadd	<db> <course> <filename> <separator> [encrypted]
fileupdate	<db> <course> <filename> <separator> [encrypted]
filedelete	<db> <course> <filename>
filechangeid	<db> <course> <filename> <separator>
homearea_xml	<db> <course> <WebCT ID> <separator pair> <server base address>

Field Name: `db`
Value: `global or student`
Example: `global`
Description: This is the name of the database, either global database or student database.

Field Name: `course`
Value: `Course ID`
Example: `cs100`
Description: - Required for student database operations.
- For global database operations, enter the placeholder value xxxx.

Field Name: `fieldsData_pair_list`
Value: A double quote-enclosed list of field-data pairs in the form: field_name=data_value.
Example: `WebCT ID=student1`
Description:
- The field names must exist in the WebCT global database or student database. The *separator* must be inserted between each of the field-data pairs.
- For the global database, the optional fields Courses and Registered Courses are available for adding and/or modifying courses and registered courses to which a global user belongs. The values for these fields can be a colon-separated list of course IDs for Courses or course names for Registered Courses. For example, `Courses=cs100:psyc100:math100.` If you also specify a user type with the course, this is separated from the course ID by a semicolon, for example, `Courses=cs100;D:psyc100;TA`. **Note**: The default user type is (S)tudent.
- A user can be added as a primary designer or as a secondary designer. The first WebCT ID added to the course as a designer becomes the primary designer; every subsequent designer becomes a secondary designer.

Field Name: fieldsData_pair_list (cont.)

Description:

- **Note**: The following are reserved words in the fieldsData_pair_list:
- Login ID (this is old terminology, and is supported for backward compatibility only. It has the same meaning as User ID).
- User ID (the User ID of a student in a course)
- Password (the password of the global user or the student)
- Global ID (this is old terminology, and is supported for backward compatibility only. It has the same meaning as WebCT ID).
- WebCT ID (WebCT ID of a global user)
- First Name (first name of the global user or the student. It is one of the reserved columns in both the global and student databases)
- Last Name (last name of the global user or the student. It is one of the reserved columns in both the global and student databases)
- Courses (the list of WebCT courses for a global user). If you populate this field through the API, the course must already exist on the WebCT server.
- Registered Courses (the list of courses maintained by the registrar for a global user. These courses may or may not have a WebCT course.)
- Thumbprint (internal data and cannot be modified)
- LockPID (internal data and cannot be modified)
- #User Type (internal data. This can be modified through the API)
- #Email (internal data and cannot be modified)
- #Password Question (internal data and cannot be modified)
- #Password Answer (internal data and cannot be modified)

Note: The reserved words are case sensitive.

Field Name: `separator`

Value: Any alphanumeric string representing the separator between data pairs in the fieldsData_pair_list.

Example: "," (comma)

Description: Delimiter used to separate data items. You must declare what value you will be using as a delimiter for the operations add, delete, changeid, update, and find.

Note: For the global database, the colon and semi-colon are not allowed as separators.

Field Name: `user_type`

Value: `user_type`

Example: `user_type`

Description: Only used with the `find` operation on the global database; return value of user_type is one of three users types, S,D,TA (for Student, Designer, Teaching Assistant)

Field Name:	`encrypted`
Value:	`encrypted`
Example:	`encrypted`
Description:	- Only used with the add, update, fileadd and fileupdate operations. - The password must be encrypted using the standard UNIX DES encryption method or the newly added or modified users may not be able to access WebCT. - Add to the end of the command line to indicate that the passwords are passing in encrypted form.

FUNCTIONS

ADDING USERS

Users can be added to the global database or student databases. However, in general, you should add users to the global database and use the Courses field to add them to each course. This method is simpler and automatically links the WebCT ID to each User ID.

ADDING A SINGLE USER TO THE GLOBAL DATABASE

Operation = add
• The fieldsData_pair_list must include both the WebCT ID and Password fields. • You can specify the user type (S for student, D for designer, TA for teaching assistant). If you don't specify a user type, the user type defaults to (S)tudent. If the user type is specified as (D)esigner and there is no existing designer, the user is added as the primary designer. If there is an existing designer, the user is added as secondary designer.

Example

Add a user named Justin Case to the global database as a designer for cs100; a teaching assistant for cs200; and as a student in cs810:

Enter the command:

```
webctdb add global xxxx "WebCT ID=jcase,Password=1234,
      First Name=Justin,Last Name=Case,Courses=cs100;D:cs200;TA:cs810;S" ","
```

ADDING A SINGLE USER TO THE STUDENT DATABASE

Operation = add
• The fieldsData_pair_list must include both the User ID and Password fields.

Example

Add a user named Bailey Wick to the student database for course cs100:

Enter the command:

```
webctdb add student cs100 "User ID=bwick,Password=1234,
      First Name=Bailey,Last Name=Wick" ","
```

ADDING MULTIPLE USERS TO THE GLOBAL DATABASE

Operation = fileadd

- filename is either a full absolute path or a relative path from the current directory to the file. For example, if the file is located in the current directory, then filename is only the name of the file. A file extension, such as `.txt`, is recommended.
- The file must be in plain text. The first line of the file must be the field names separated by the separator string. The rest of the file contains the data, one record per line. Data should be in the same order as the field names, separated by the value of the separator. There must be no spaces between the data and the separators.
- If the user exists in the database, `fileadd` will send an error message to STDOUT. The user record will not be changed; the process will skip to the next record in the file.
- An optional `encrypted` argument can be added at the end of the command line to indicate that the passwords are passing in an encrypted form. The passwords should be encrypted using the standard UNIX DES encryption method or the newly added or modified users may not be able to access WebCT.

Example

Add users to the global database from a text file named `users.txt`.

SAMPLE USERS.TXT FILE:

```
WebCT ID,Password,Last Name,First Name
jsmith,9876,Smith,John
jbrown,2345,Brown,Jane
bfawlty,8765,Fawlty,Basil
arigsby,5432,Rigsby,Arthur
```

Enter the command:

```
webctdb fileadd global xxxx users.txt ","
```

ADDING MULTIPLE USERS TO THE STUDENT DATABASE

Operation = fileadd

- filename is either a full absolute path or a relative path from the current directory to the file. For example, if the file is located in the current directory, then filename is only the name of the file. A file extension, such as `.txt`, is recommended.
- The file must be in plain text. The first line of the file must be the field names separated by the separator string. The rest of the file contains the data, one record per line. Data should be in the same order as the field names, separated by the value of the separator. There must be no spaces between the data and the separators.
- If the user exists in the database, `fileadd` will send an error message to STDOUT. The user record will not be changed in the database; the process will skip to the next record in the file.
- An optional `encrypted` argument can be added at the end of the command line to indicate that the passwords are passing in an encrypted form. The passwords should be encrypted using the standard UNIX DES encryption method or the newly added users may not be able to access WebCT.

Example

Add students whose records are stored in the file `class.txt` to the course cs100.

SAMPLE CLASS.TXT FILE

```
User ID,Password,Last Name,First Name,
jsmith,9876,Smith,John
jbrown,2345,Brown,Jane
bfawlty,8765,Fawlty,Basil
arigsby,5432,Rigsby,Arthur
```

Enter the command:

```
webctdb fileadd student cs100 class.txt ","
```

UPDATING USERS

UPDATING A SINGLE USER IN THE GLOBAL DATABASE

Operation = update

- The fieldsData_pair_list must include the WebCT ID.
- Empty fields are not changed.
- If the field value is "_DELETE_", the value will be set to null
- The Standard API behavior when updating the Courses and Registered Courses field is to always overwrite the field. If you supply a Courses field in your update, the user's WebCT ID will be linked to the courses that you supply, and unlinked from any pre-existing courses that you do not supply.
- You can update a user type by specifying a different one.
- `update` cannot be used to modify quiz scores.
- An optional `encrypted` argument can be added at the end of the command line to indicate that the passwords are passing in an encrypted form. The passwords should be encrypted using the standard UNIX DES encryption method or updated users may not be able to access WebCT.

Example

For the student Justin Case, password 1234, with the following courses: cs100(D) cs200(TA) cs810(S), update the password in the global database and update the courses so that only cs100 remains.

Enter the command:

```
webctdb update global xxxx "WebCT ID=jcase,Password=abcd,
      Courses=cs100" ","
```

UPDATING A SINGLE USER IN THE STUDENT DATABASE

Operation = update

- The fieldsData_pair_list must include the User ID field.
- Empty fields are not changed.
- If the field value is "_DELETE_", the value will be set to null
- The Standard API behavior when updating the Courses and Registered Courses field is to always overwrite the field. If you supply a Courses field in your update, the user's WebCT ID will be linked to the courses that you supply, and unlinked from any pre-existing courses that you do not supply
- You can update a user type by specifying a different one.
- `update` cannot be used to modify quiz scores.
- An optional `encrypted` argument can be added at the end of the command line to indicate that the passwords are passing in an encrypted form. The passwords should be encrypted using the standard UNIX DES encryption method or the updated users may not be able to access WebCT.

Example

To update the student Bailey Wick, first name, last name, and password of password "1234".

Enter the command:

```
webctdb update student cs100 "User ID=bwick,Password=abcd,
      First Name=Bailie,Last Name=Wicke" ","
```

UPDATING MULTIPLE USERS IN THE GLOBAL DATABASE

Operation = fileupdate

- filename is either a full absolute path or a relative path from the current directory to the file. For example, if the file is located in the current directory, then filename is the name of the file. A file extension, such as `.txt`, is recommended.
- The file must be in plain text. The first line of the file must be the field names separated by the separator string. The rest of the file contains the data, one record per line. Data should be in the same order as the field names, separated by the value of the separator. There must be no spaces between the data and the separators.
- An optional `encrypted` argument can be added at the end of the command line to indicate that the passwords are passing in an encrypted form. The passwords should be encrypted using the standard UNIX DES encryption method or the newly added or modified users may not be able to access WebCT.
- `fileupdate` will add a user if they do not exist in the database.
- Empty fields will not be changed.
- For `fileupdate`, if the value field value is "_DELETE_", the value will be set to null
- The Standard API behavior when updating the Courses and Registered Courses field is to always overwrite the field. If you supply a Courses field in your update, the user's WebCT ID will be linked to the courses that you supply, and unlinked from any pre-existing courses that you do not supply

Example

Change the names of a group of users whose updates are contained in the file `updates.txt`.

SAMPLE UPDATES.TXT FILE:

```
WebCT ID,Last Name,First Name
jsmith,Smith,Jerry
jbrown,Brown,Janet
bfawlty,Fawlty,Brian
arigsby,Rigsby,Alan
```

Enter the command:

```
webctdb fileupdate global xxxx updates.txt ","
```

UPDATING MULTIPLE USERS IN THE STUDENT DATABASE

Operation = fileupdate

- filename is either a full absolute path or a relative path from the current directory to the file. For example, if the file is located in the current directory, then filename is the name of the file. A file extension, such as `.txt`, is recommended.
- The file must be in plain text. The first line of the file must be the field names separated by the separator string. The rest of the file contains the data, one record per line. Data should be in the same order as the field names, separated by the value of the separator. There must be no spaces between the data and the separators.
- An optional `encrypted` argument can be added at the end of the command line to indicate that the passwords are passing in an encrypted form. The passwords should be encrypted using the standard UNIX DES encryption method or the newly added or modified users may not be able to access WebCT.
- `fileupdate` will add a student or user if they do not already exist in the database.
- Empty fields will not be changed.
- For `fileupdate`, if the field value is "_DELETE_", the value will be set to null
- `fileupdate` overwrites the data fields being changed; it does not append.

Example

Change the names of students in the course cs100 using updates contained in the file `updates.txt`.

SAMPLE UPDATES.TXT FILE:

```
User ID,Last Name,First Name
jsmith,Smith,Jerry
jbrown,Brown,Janet
bfawlty,Fawlty,Brian
arigsby,Rigsby,Alan
```

Enter the command:

```
webctdb fileupdate student cs100 updates.txt ","
```

DELETING USERS

DELETING A SINGLE USER FROM THE GLOBAL DATABASE

Operation = delete

- global_id is the ID of the user to be deleted from the global database.

Note: Depending on the *User Data* setting in the administrator interface, the student's data may also be deleted from the student database.

Example

Delete the global database record for the user whose WebCT ID is jcase. **Note**: The student will be denied access to all the courses listed in their global database record. Depending on the *User Data* setting in the administrator interface, the student's data may also be deleted from the student database.

Enter the command:

```
webctdb delete global xxxx jcase
```

DELETING A SINGLE USER FROM THE STUDENT DATABASE

Operation = delete
• user_id is the ID of the student to be deleted from the student database.

Example

Delete the record for the student in the cs100 course whose User ID is bwick.

Enter the command:

```
webctdb delete student cs100 bwick
```

DELETING MULTIPLE USERS FROM THE GLOBAL DATABASE

Operation = filedelete
• filename is the either a full absolute path or a relative path from the current directory to the file. For example, if the file is located in the current directory, then filename is simply the name of the file. A file extension, such as `.txt`, is recommended. • If the user does not exist in the database, `filedelete` will send an error message to STDOUT. The process will skip to the next record in the file.

Example

Delete users from the global database using a text file `deleteusers.txt`.

SAMPLE DELETEUSERS.TXT FILE:

```
jsmith
jbrown
bfawlty
arigsby
```

Enter the command:

```
webctdb filedelete global xxxx deleteusers.txt ","
```

DELETING MULTIPLE USERS FROM THE STUDENT DATABASE

Operation = filedelete

Operation = filedelete

- filename is the either a full absolute path or a relative path from the current directory to the file. For example, if the file is located in the current directory, then filename is simply the name of the file. A file extension, such as `.txt`, is recommended.
- If the user does not exist in the database, `filedelete` will send an error message to STDOUT The process will skip to the next record in the file.

Example

Delete students whose records are stored in the file `delete.txt` from the course cs100.

SAMPLE DELETE.TXT FILE:

```
jsmith
jbrown
bfawlty
arigsby
```

Enter the command:

```
webctdb filedelete student cs100 delete.txt
```

FINDING WUUIS

Important: Since the release of WebCT 3.6, the use of the WUUI for Automatic Signon and the `find_wuui` operation are deprecated. With WebCT moving towards the use of the IMS specifications, which are becoming standards in the learning community, the IMS ID and IMS source are now preferred over the WUUI. Although the use of the WUUI is deprecated, the function will still be supported for 3.7.

FINDING WUUIS USING IMS IDS

Operation = find_ims_id_wuui

- The WUUI (WebCT Unique Universal Identifier) is a 32-character alphanumeric string that identifies a global user in WebCT.
- The `find_ims_id_wuui` operation is for the global database only.
- This operation is similar to the `find_wuui` operation, except that your campus portal passes the user's IMS ID, not their WebCT ID. WebCT returns the user's WUUI.
- This operation can be used only when WebCT's global database has been populated using the IMS Enterprise API, because only in those cases would an IMS ID be present for each user in the WebCT global database.
- IMS ID is the IMS ID of the user in the global database.
- The WUUI is sent to STDOUT.

Example

Find the WUUI for the user whose IMS ID is jcase.

Enter the command:

```
webctdb find_ims_id_wuui global xxxx jcase
```

The server returns:

```
Success: WUUI=abcdefghijklmnopqrstuvwxyz123456
```

FINDING USERS

FINDING A USER IN THE GLOBAL DATABASE

Operation = find
• WebCT ID is the WebCT ID of the user in the global database. • Separator is the separator of the output data, which is sent to STDOUT in the same format as the fieldsData_pair_list. • If the field name user_type is specified in a global database query, the user type (S,D,TA) will be included in the result.

Example

Find the global database record, including user type, for the user with the WebCT ID jcase.

Enter the command:

```
webctdb find global xxxx jcase "," user_type
```

If the command is successfully executed:

```
Success: WebCT ID=jcase,First Name=Justin,
      Last Name=Case,Courses=cs100;D:cs200;TA:cs810;S
```

FINDING A USER IN THE STUDENT DATABASE

Operation = find
• user_id is the User ID of the student in the student database. • Separator is the separator of the output data, which is sent to STDOUT in the same format as the fieldsData_pair_list.

Example

Find the student in the cs100 course whose User ID is bwick.

Enter the command:

```
webctdb find student cs100 bwick ","
```

If the command is successfully executed:

```
Success:First Name=Bailie,Last Name=Wicke,User ID=bwick
```

CHANGING WEBCT IDS

CHANGING A SINGLE USER'S WEBCT ID

Operation = changeid

Operation = changeid
• `changeid` can only be used on the global database. • old_id is the WebCT ID to be changed. • new_id is the new WebCT ID.

Example

Change Justin Case's WebCT ID from jcase to jicase.

Enter the command:

```
webctdb changeid global xxxx "Old ID=jcase,New ID=jicase" ","
```

CHANGING MULTIPLE USERS' WEBCT IDS

Operation = filechangeid
• filename is either a full absolute path or a relative path from the current directory to the file. For example, if the file is located in the current directory, then filename is simply the name of the file. A file extension, such as `.txt`, is recommended. • The first line of the data file should be the field names Old ID and New ID, separated by the separator string. The rest of the file contains the data, one record per line. Data should be in the same order as the field names, separated by the value of the separator. **Note**: The field name Old ID does not exist in the databases. • If the user does not already exist in the database, `filechangeid` will send an error message to STDOUT The process will skip to the next record in the file.

Example

Change the WebCT IDs of a group of users contained in a file `changeusers.txt`.

SAMPLE CHANGEUSERS.TXT FILE:

```
Old ID,New ID
jsmith,jtsmith
jbrown,jkbrown
bfawlty,befawlty
arigsby,aurigsby
```

Enter the command:

```
webctdb filechangeid global xxxx changeusers.txt ","
```

EXPORTING MYWEBCT IN XML FORMAT

This Standard API command exports a user's *myWebCT* in XML format, which allows *myWebCT* information to be modified and redisplayed in a desired format. For example, the information could be integrated with a campus portal.

This command can be used in conjunction with automatic signon, allowing for a single point of authentication, see *Chapter 2: Automatic Signon From Other Systems*.

Operation = homearea_xml
• `homearea_xml` can only be used on the global database. • WebCT ID is the WebCT ID of the user whose *myWebCT* you want to export in XML format. • Separator is the separator of the output data, which is sent to STDOUT in the same format as the fieldsData_pair_list. • server base address is the address of the WebCT server.

The XML that is returned is compliant with the DTD located in `<install_dir>/webct/webct/generic/api/xml/webct2.dtd`

The XML can be parsed to extract the required elements. Link elements that require authentication by WebCT contain the attribute "secure" with a value of TRUE.

Example

Export *myWebCT* in XML format for the user whose WebCT ID is `jsmith` and whose server base address is `http://webctserver:port`.

Enter the command:

```
webctdb homearea_xml global xxxx jsmith "," http://webctserver:port
```

WEB-BASED INTERFACE (SERVE_WEBCTDB)

The Web-based Standard API allows data in the WebCT global database and student databases to be queried and manipulated by remote servers. For example, the Web-based interface could be used to make changes to global database records based on registration changes driven by events on another system. It could also be used to create a custom administrator's interface.

Implementing the Web-based interface involves two steps.

1. Setting the API shared secret value
2. Developing a program to generate an HTTP request

Step 1 can be accomplished by a WebCT administrator who has basic knowledge of the WebCT file system. Step 2 requires an experienced Web developer.

1. SETTING THE API SHARED SECRET VALUE

The shared secret value is a key component of allowing external servers to automatically sign on users to WebCT. The shared secret value is used to create a Message Authentication Code (MAC) from the submitted data. When WebCT receives a request, it decodes the shared secret value from MAC using the submitted data. If the decoded shared secret value is the same as the one stored locally, the request is considered authentic and is processed. You can set the shared secret value by performing the following steps:

1. Using a text editor, open the file `<webct_install_directory>/webct/webct/generic/api/api_secret`
2. Change the first line of the file to your desired secret. (For security reasons, the default value "SECRET" does not work). You should note the following about the shared secret value.
   - It cannot exceed 256 characters.
   - It cannot contain tab, or other control characters.
   - It should not contain end-of-line characters. **Note**: By default, the UNIX text editor vi and pico automatically add end-of-line characters. Check the file size to ensure that the number of characters equals the number of bytes.
   - It is case-sensitive
3. Save the file.

Because the shared secret value has such a critical role, choose it carefully.

Tips for Shared Secrets

- Make your shared secret value difficult to guess by making it lengthy and by including a combination of numbers and upper and lower case characters.
- Change your shared secret value at regular intervals.
- On remote systems, place shared secret values in secure directories.

2. Developing a Program to Generate an HTTP Request

Developing a program to generate an http request is the most substantive part of implementing the Web-based standard API. The program must:

- Generate a Message Authentication Code (MAC)
- Assemble a properly formatted http request
- Process any data being returned

Creating Message Authentication Codes

Because the Web interfaces to the Standard API and Automatic signon reside in public directories, Message Authentication Codes (MACs) are required to ensure that only messages from trusted servers are processed.

WebCT provides three options to assist you in creating MACs:

1. A C function that you may integrate and compile into your C program
2. An executable file to which you make a system call from your program
3. Instructions for generating a MAC using a language of your choice

Option 1: Using the get_authentication C Function

The get_authentication function generates a MAC from an array of data and a shared secret value.

The source code necessary to use the C function is located in `<webct_install_directory>/webct/webct/generic/api/security/`

A test program, which contains a Makefile for UNIX based systems, is also provided.

The file `api_security.c` contains the `get_authentication` function.

get_authentication	Generates a MAC from an array of data and a shared secret value
Syntax	char* get_authentication (int i, char* data[], char* secret, char* encrypted_data)
Returns	32 byte alphanumeric MAC

Parameter	Description
i	The number of elements in the array data[].
data	Array of all values to be used in generating the MAC. The data should not be URL encoded.
secret	The shared secret value.
encrypted_data	The memory location of the MAC. It must be at least 32 bytes long.

OPTION 2: USING THE MESSAGE AUTHENTICATION CODE GENERATOR (GET_MD5 EXECUTABLE)

The Message Authentication Code (MAC) generator generates a MAC from a shared secret value and a string consisting of the IMS ID, a timestamp, and a destination URL.

Use the Message Authentication Code generator (an executable called get_md5) if you are not working in C, or do not want to create a function to create the MAC. You can make a system call to get_md5 from your program and have the authentication string returned. The get_md5 executable has no dependencies on WebCT and can be copied to other servers as required. If you need a get_md5 executable for an operating system other than the one your WebCT server is running on, you can download several pre-compiled binaries for other operating systems on WebCT.com.

get_md5	Generates a MAC from a shared secret value and a line of data
Syntax	get_md5 <shared_secret_filename> <data_to_encrypt>
Returns	32 byte alphanumeric MAC

Attribute	Description
shared_secret_filename	The filename where the shared secret value is stored.

`string_to_encrypt`	The string to be encrypted. The string should not be URL encoded.

OPTION 3: CREATE A MAC USING A LANGUAGE OF YOUR OWN CHOICE

If you want to create MACs within your code, (e.g. you are writing your code in Java and don't want to make a system call), you can create a MAC with the following procedure:

1. Calculate the total of the ASCII values of all the characters in the data.
2. Convert the total of the ASCII values into a string.
3. Append the shared secret value.
4. Encrypt the string into a 16-byte string using the MD5 algorithm.
5. Convert the 16-byte string into a 32-byte alphanumeric string to make it URL-friendly.

ASSEMBLING THE HTTP REQUEST

There are several options for assembling an http request to the Web-based standard API. The option you choose will be based on your programming language of choice, and how you want to communicate with the Web server. You can issue API commands in several ways, including:

- Socket programming directly with the Web server
- Using a library which simulates a user agent
- Assembling a GET request and refreshing a browser window with the query string.

In Perl, you have the option of communicating directly with the Web server using the IO::Socket module included with most basic distributions, or installing and using a module such as LWP which simulates a user agent (e.g. a Web browser). Similar modules are available for most popular languages such as C or Java.

If you wish to refresh a user's browser window with a query string, you can do so using the "Location" http header, HTML meta tags, or using JavaScript's location.replace method.

SYNTAX

The general syntax for a Web-based request to the Standard API is as follows:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=<operation>&DB=<db>
      &COURSE=<course_id | placeholder>&AUTH=<32_byte_mac>
      [&User%20ID=<user_id> | &WebCT%20ID=<webct_id>][&IMS%20ID=<ims_id>]
      [&USER%20TYPE=<1_or_0>][&ENCRYPTED=<1_or_0>][&field1=<field1>]
      [&fieldn=<fieldn>]HTTP/1.0
```

where:

Key	Value	Description
OPERATION	add	Adds a user to the global or student database. If the user already exists, an error is returned.
	update	Updates an existing user in the global or student database. If the user does not exist, this operation returns an error.

Key	Value	Description
	delete	Deletes a single user from the global or student database.
	find_wuui	Finds the WUUI for a user using their WebCT ID as the key.
	find_ims_id_wuui	Finds the WUUI for a user using their IMS ID (Person→SourcedID→ID) as the key.
	find	Finds the user record based on the User ID (if searching the student database) or WebCT ID (if searching the global database).
	changeid	Changes a WebCT ID.
	homearea_xml	Exports a user's *myWebCT* in XML format.

Notes:

- The Standard API can accept GET or POST requests. POST requests can put their key/value pairs in the query string or in the body of the message in the appropriate format (see the W3C HTML 4.01 Specification at http://www.w3.org/TR/html401/interact/forms.html#h-17.13.4)
- Requests must be URL encoded (e.g. spaces should be replaced with %20)
- Key/value pairs may appear in any order
- Syntax examples represent http requests directly to the Web server. If you are using a programming module to create your requests (such as LWP in Perl), many details of the request may be transparent to you.

FUNCTIONS

ADDING USERS

ADDING A USER TO THE GLOBAL DATABASE

Add operations have the following syntax:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=add&DB=global
      &COURSE=<placeholder>&AUTH=<32_byte_mac>&WebCT%20ID=<webct_id>
      &Password=<password>[&field1=<field1>][&fieldn=<fieldn>]
      [&ENCRYPTED=<1_or_0>]HTTP/1.0
```

where:

Key	Value	Description
COURSE	Any alphanumeric string	This is a required placeholder value. You may use any alphanumeric value, but ensure that you use it in the calculation of the MAC.
AUTH	32 byte MAC	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.

Key	Value	Description
WebCT ID	WebCT ID	The WebCT ID of the user being added. WebCT IDs can contain alphanumeric strings, underscores, and periods.
Password	Password	The password to be used for the user being added. Passwords can consist of any alphanumeric string. The API does not enforce minimum password lengths.
Field1 … Fieldn (optional)	Data associated with the column specified as a key.	Any of the default columns within the global database (First Name, Last Name, Courses, Registered Courses) can be modified using the syntax ColumnName=Value. Administrator-created columns can also be modified using this method.
ENCRYPTED (optional)	1	Enables pre-encrypted password support. With the encrypted argument set to 1, you should pass passwords encrypted with the standard UNIX DES method when using this setting.
	0 (default)	Disables pre-encrypted password support (default). In this mode, passwords should be submitted as clear-text.

Note: The Courses field uses a colon as a delimiter between courses, and a semicolon as a delimiter between user types. Thus the string "HKIN100;D:HKIN200;TA:HKIN300;S" indicates that a user is to be added to HKIN100 as a designer, HKIN200 as a teaching assistant and HKIN300 as a student. If no user type is specified, WebCT will default to adding the user as a student. Similarly, the Registered Courses field is colon delimited. See the System Administrators Guide for more information on the Courses and Registered courses field.

Example

Add a user to the global database, and enroll them in the course ENGL100 as a designer, ENGL560 as a student, and ENGL477 as a teaching assistant.

```
GET  /webct/public/serve_webctdb?OPERATION=add&DB=global&COURSE=xxxx
      &AUTH= EB1A09F0BB299C23E99A5978587F49C1&WEBCT%20ID=pinto
      &PASSWORD=an1mal&FIRST%20NAME=Larry&LAST%20NAME=Kroger&
      COURSES=ENGL100;D:ENGL560:ENGL477;TA HTTP/1.0
```

ADDING A USER TO THE STUDENT DATABASE

Students can be added to the student database using the following syntax:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=add&DB=student
      &COURSE=<course_id>&AUTH=<32_byte_mac>&User%20ID=<user_id>
      &Password=<password>[&field1=<field1>][&fieldn=<fieldn>]
      [&ENCRYPTED=<1_or_0>]HTTP/1.0
```

where:

Key	Value	Description
COURSE	WebCT Course ID	The WebCT course to which the user will be added.
AUTH	32 byte MAC	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
User ID	User ID	The User ID of the user being added.
Password	Password	The password to be used for the user being added. The API does not enforce minimum password lengths.
Field1 … Fieldn (optional)	Data associated with the column specified as a key.	Any of the default columns within the global database (First Name, Last Name, Courses, Registered Courses) can be modified using the syntax ColumnName=Value. Administrator-created columns can also be modified using this method.
ENCRYPTED (optional)	1	Enables pre-encrypted password support. With the encrypted argument set to 1, you should pass passwords encrypted with the standard UNIX DES method when using this setting
	0 (default)	Disables pre-encrypted password support (default). In this mode, passwords should be submitted as clear

Example

Add a student to the student database of the course ENGL588. In addition, add data to a pre-existing column "StudentNumber" (This is a custom column created by the designer). Because this user is being added to the student database only, they are considered an "orphan user" until a WebCT ID is associated with this User ID:

```
GET  /webct/public/serve_webctdb?OPERATION=add&DB=student
      &COURSE=ENGL588&AUTH=EB1A09F0BB299C23E99A5978587F49C1
      &User%20ID=flounder&Password=an1mal&First%20Name=Kent
      &Last%20Name=Dorfman&StudentNumber=123456789 HTTP/1.0
```

UPDATING USERS

UPDATING A USER IN THE GLOBAL DATABASE

Updating users in the global database is very similar to adding users. The syntax is as follows:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=update&DB=global
      &COURSE=<placeholder>&AUTH=<32_byte_mac>&WebCT%20ID=<webct_id>
      [&FIELD1=<field1>][&FIELDN=<fieldn>][&ENCRYPTED=<1_or_0>]HTTP/1.0
```

where:

Key	Value	Description
COURSE	Any alphanumeric string	This is a required placeholder value. You may use any alphanumeric value, but ensure that you use it in the

Key	Value	Description
		calculation of the MAC.
AUTH	32 byte MAC	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
WebCT ID	Existing WebCT ID	The WebCT ID of the user being added. WebCT IDs may contain alphanumeric strings, underscores, and periods.
Password (optional)	Password	The password to be used for the user being updated. The API does not enforce minimum password lengths.
Field1 … Fieldn (optional)	Data associated with the column specified as a key.	Any of the default columns within the global database (First Name, Last Name, Courses, Registered Courses) can be modified using the syntax ColumnName=Value. Administrator-created columns can also be modified using this method.
	DELETE	The "_DELETE_" keyword deletes the data from the field specified in the key and sets it to undefined.
ENCRYPTED (optional)	1	Enables pre-encrypted password support. With the encrypted argument set to 1, you should pass passwords encrypted with the standard UNIX DES method when using this setting
	0 (default)	Disables pre-encrypted password support (default). In this mode, passwords should be submitted as clear-text.

Notes:

- The *User Data* setting in the WebCT Administrator interface affects how updating the Courses column will modify the student database when unlinking WebCT IDs from User IDs. If the User Data setting is selected, user data is left in the student database.
- The Standard API always overwrites the Courses and Registered Course fields when updating. If you supply a Courses field in your update, the user's WebCT ID will be linked to the courses that you specify, and unlinked from any pre-existing courses that you do not specify.
- The Courses field uses a colon as a delimiter between courses, and a semicolon as a delimiter between user types. Thus the string "HKIN100;D:HKIN200;TA:HKIN300;S" indicates that a user is to be added to HKIN100 as a designer, HKIN200 as a teaching assistant and HKIN300 as a student. If no user type is specified, WebCT will default to adding the user as a student. Similarly, the Registered Courses field is colon delimited. See the *WebCT 3.7 Campus Edition System Administrator's Guide* for more information on the Courses and Registered courses field.

Example

A user is currently enrolled in three courses: ENGL101 as a designer, ENGL560 as a student, and ENGL477 as a teaching assistant. This example unlinks the WebCT ID from the User ID for ENGL 560 and ENGL 477, and adds the WebCT ID to the course ENGL101 as designer.

```
GET  /webct/public/serve_webctdb?OPERATION=update&DB=global&COURSE=xxxx&AUTH=EB1A
     09F0BB299C23E99A5978587F49C1&WebCT%20ID=pinto&Courses=ENGL101;D:ENGL101;D
     HTTP/1.0
```

The user is unlinked from the two courses because API updates always overwrite fields.

UPDATING A USER IN THE STUDENT DATABASE

Updating students in the student database is very similar to adding students. The syntax is as follows:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=update&DB=student
     &COURSE=<course_id>&AUTH=<32_byte_mac>&User%20ID=<user_id>
     [&field1=<field1>][&fieldn=<fieldn>][&ENCRYPTED=<1_or_0>]HTTP/1.0
```

where:

Key	Value	Description
COURSE	WebCT Course ID	The WebCT course in which the user's data is updated.
AUTH	32 byte MAC	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
User ID	User ID	The User ID of the user being added.
Password (optional)	Password	The password to be used for the user being added. The API does not enforce minimum password lengths.
Field1 … Fieldn (optional)	Data associated with the column specified as a key.	Any of the default columns within the global database (First Name, Last Name, Courses, Registered Courses) can be modified using the syntax ColumnName=Value. Administrator-created columns can also be modified using this method.
	DELETE	The "_DELETE_" keyword deletes the data from the field specified in the key and sets it to undefined.
ENCRYPTED (optional)	1	Enables pre-encrypted password support. With the encrypted argument set to 1, you should pass passwords encrypted with the standard UNIX DES method when using this setting.
	0 (default)	Disables pre-encrypted password support (default). In this mode, passwords should be submitted as clear-text.

Example

In the following example, a student record is updated with information for the instructor-added numeric columns "Student Participation" and "Bonus" in the course MATH100.

```
GET /webct/public/serve_webctdb?OPERATION=update&DB=student&
     COURSE=MATH100&AUTH=EB1A09F0BB299C23E99A5978587F49C1
     &User%20ID=otter&Student%20Participation=100&Bonus=34 HTTP/1.0
```

DELETING USERS

DELETING A USER FROM THE GLOBAL DATABASE

The syntax for deleting a user from the global database is as follows:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=delete&DB=global
     &COURSE=<placeholder>&AUTH=<32_byte_mac>&WebCT%20ID=<webct_id>
     HTTP/1.0
```

where:

Key	Value	Description
COURSE	Any alphanumeric string	This is a required placeholder value. You can use any alphanumeric value, but ensure that you use it in the calculation of the MAC.
AUTH	32 byte MAC	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
WebCT ID	WebCT ID	The WebCT ID of the user being deleted.

Note: The *User Data* setting in the WebCT Administrator interface affects whether user data is left in a course when a user record is deleted from the global database. If the *User Data* setting is selected, user data is left in the student database.

Example

In this example, the user record for the user with the WebCT ID "neidermeyer" is deleted from the global database:

```
GET  /webct/public/serve_webctdb?OPERATION=delete&DB=global&COURSE=xxxx
     &AUTH=EB1A09F0BB299C23E99A5978587F49C1&WebCT%20ID=neidermeyer
     HTTP/1.0
```

DELETING A USER FROM THE STUDENT DATABASE

The syntax for deleting a student from the student database is as follows:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=delete&DB=student
     &COURSE=<course_id>&AUTH=<32_byte_mac>&User%20ID=<user_id>
     HTTP/1.0
```

where:

Key	Value	Description
COURSE	WebCT Course ID	The WebCT course from which the user will be deleted.

AUTH	32 byte MAC	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
User ID	User ID	The User ID of the user being deleted.

Example

In this example, the student with the User ID "stork" is deleted from the course PSYCH204-23:

```
GET  /webct/public/serve_webctdb?OPERATION=delete&DB=student
      &COURSE=PSYCH204-23&AUTH=EB1A09F0BB299C23E99A5978587F49C1
      &User%20ID=stork HTTP/1.0
```

FINDING WUUIS

Important: Since the release of WebCT 3.6, the use of the WUUI for Automatic Signon and the find_wuui operation are deprecated. With WebCT moving towards the use of the IMS specifications, which are becoming standards in the learning community, the IMS ID and IMS source are now preferred over the WUUI. Although the use of the WUUI is deprecated, the functionality remains.

The syntax for finding WUUIs is as follows:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=<operation>
      &DB=global&field1=<field1>&COURSE=<placeholder>
      &AUTH=<32_byte_mac> HTTP/1.0
```

Key	Value	Description
OPERATION	find_wuui	Finds a user's WUUI from a WebCT ID
	find_ims_id_wuui	Finds a user's WUUI from an IMS ID
AUTH	32 byte MAC	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
field1	WebCT ID	Use if the operation is find_wuui
	IMS ID	Use if the operation is find_ims_id_wuui
COURSE	Any alphanumeric string	This is a required placeholder value. You can use any alphanumeric value, but ensure that you use it in the calculation of the MAC.

EXAMPLES

FINDING A USER'S WUUI FROM A WEBCT ID

Find the WUUI for the WebCT ID "jdoe".

```
GET  /webct/public/serve_webctdb?OPERATION=find_wuui&DB=global
      &WebCT%20ID=jdoe&COURSE=xxxx&AUTH=EB1A09F0BB299C23E99A5978587F49C1
      HTTP/1.0
```

The Web server returns the following (not including http headers):

```
Success: #WUUI = 6321BB2537BE7F1E26375D4E1687EE1F
```

FINDING A USER'S WUUI FROM AN IMS ID

Find the WUUI for the IMS ID (Person→SourcedID→ID) "123456789":

```
GET  /webct/public/serve_webctdb?OPERATION=find_ims_id_wuui&DB=global
      &IMS%20ID=123456789&COURSE=xxxx&
      AUTH=EB1A09F0BB299C23E99A5978587F49C1 HTTP/1.0
```

The Web server returns the following (not including http headers):

```
Success: #WUUI = 6321BB2537BE7F1E26375D4E1687EE1F
```

FINDING USERS

FINDING A USER IN THE GLOBAL DATABASE

To find a user's global database record, the following syntax is used:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=find&DB=global
      &COURSE=<placeholder>&AUTH=<32_byte_mac>&WebCT%20ID=<webct_id>
      [USER%20TYPE=<1_or_0>]HTTP/1.0
```

where:

Key	Value	Description
OPERATION	find	Finds the user record for a given WebCT ID
COURSE	Any alphanumeric string	This is a required placeholder value. You can use any alphanumeric value, but ensure that you use it in the calculation of the MAC.
AUTH	32_byte_mac	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
WebCT ID	WebCT_ID	The WebCT ID of the record you want to display.
USER TYPE (optional)	1	With the User Type option enabled, the global database record generated includes user type information that indicates whether a user is a designer, student, or teaching assistant for the course.
	0 (default)	No user type information is generated.

Example

In this example, the complete record including user type information is returned for the user with the WebCT ID “pinto”, who is enrolled in three courses.

```
GET  /webct/public/serve_webctdb?OPERATION=find&DB=global&COURSE=xxxx
      &AUTH=EB1A09F0BB299C23E99A5978587F49C1&WebCT%20ID=pinto
      &USER%20TYPE=1 HTTP/1.0
```

The Web server returns the following, not including http headers:

```
Success: WebCT ID=pinto,First Name=Larry,Last Name=Kroger,Courses=
      ENGL100;D:ENGL560;S:ENGL477;TA
```

FINDING A USER IN THE STUDENT DATABASE

To find a student’s student database record, the following syntax is used:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=find&DB=student
      &COURSE=<course_id>&AUTH=<32_byte_mac>&User%20ID=<user_id> HTTP/1.0
```

where:

Key	Value	Description
OPERATION	find	Finds a user’s record for a given WebCT ID.
COURSE	Any alphanumeric string	The course that you are searching.
AUTH	32_byte_mac	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
User ID	User ID	The User ID of the record you wish to display.

Example

In this example, a complete student database record is displayed for the user with User ID “chip” in the course “HKIN455”:

```
GET  /webct/public/serve_webctdb?OPERATION=find&DB=student
      &COURSE=HKIN455=AUTH=EB1A09F0BB299C23E99A5978587F49C1&User%20ID=chip
      HTTP/1.0
```

The Web server returns the following, not including http headers:

```
Success: First Name=Chip,Last Name=Diller,User ID=chip,Quiz1=36,Assignment1=10
```

CHANGING WEBCT IDS

CHANGING A USER'S WEBCT ID

To change a WebCT ID for a user, use the syntax:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=changeid&DB=global
      &COURSE=<placeholder>&AUTH=<32_byte_mac>&Old%20ID=<old_webct_id>
      &New%20ID=<new_webct_id> HTTP/1.0
```

where:

Key	Value	Description
OPERATION	changid	Changes the WebCT ID of a user
COURSE	Any alphanumeric string	This is a required placeholder value. You can use any alphanumeric value, but ensure that you use it in the calculation of the MAC.
AUTH	32_byte_mac	This is the 32 byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.
Old ID	Old WebCT ID	The WebCT ID of the record you want to change.
New ID	New WebCT ID	The WebCT ID that you want to assign to the user.

Example

In this example, the WebCT ID "flounder" is changed to "dorfmank":

```
GET  /webct/public/serve_webctdb?OPERATION=changeid&DB=global&COURSE=xxxx
      &AUTH=EB1A09F0BB299C23E99A5978587F49C1&Old%20ID=flounder
      &New%20ID=dorfmank HTTP/1.0
```

The Web server returns the following, not including http headers:

```
Success:
```

EXPORTING MYWEBCT IN XML FORMAT

This Standard API command exports a user's *myWebCT* in XML format, which allows *myWebCT* information to be modified and redisplayed in a desired format. For example, the information could be integrated with a campus portal.

This command can be used in conjunction with automatic signon, allowing for a single point of authentication, see *Chapter 2: Automatic Signon From Other Systems*.

To export a user's *myWebCT* in XML format, use the syntax:

```
<GET | POST>  /webct/public/serve_webctdb?OPERATION=homearea_xml&DB=global
      &WebCT%20ID=<WebCT ID>&AUTH=<32_byte_mac>
```

where:

Key	Value	Notes
OPERATION	homearea_xml	Exports a user's *myWebCT* in XML format.
WebCT ID	WebCT ID	The WebCT ID of the user whose *myWebCT* you want to export in XML format.
AUTH	32_byte_mac	This is the 32-byte hexadecimal string generated using the get_authentication C code, the get_md5 program, or using custom code.

The XML that is returned is compliant with the DTD located in
`<install_dir>/webct/webct/generic/api/xml/webct2.dtd`

The XML can be parsed to extract the required elements. Link elements that require authentication by WebCT contain the attribute "secure" with a value of `TRUE`.

RESOURCES

LDAP RESOURCES

WEB SITES

LDAP Guru (http://www.ldapguru.com)
Provides a large database of articles and resources

Open LDAP (http://www.openldap.org)
Home of the OpenLDAP Project

iPlanet Directory Server (http://developer.iplanet.com/tech/directory/)
Information on iPlanet, one of the most commonly used LDAP servers.

KERBEROS RESOURCES

WEB SITES

Kerberos: The Network Authentication Protocol
(http://web.mit.edu/kerberos/www/)
The home of the free MIT Kerberos implementation

The Kerberos FAQ
(http://www.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html)
Updated monthly and has answers to many common questions

Windows 2000 Kerberos Authentication
(http://www.microsoft.com/windows2000/techinfo/howitworks/security/kerberos.asp)
Detailed document about how Kerberos works in a Windows 2000 environment

IMS RESOURCES

WEB SITES

The IMS Enterprise Specifications Site
(http://www.imsproject.org/enterprise/)
The authoritative resource for the IMS Enterprise Information model, the XML Binding Specification and the Best Practices and Implementation Guide