Blackboard

Blackboard Academic Suite™

Advanced Integration and Data Management

Blackboard Academic Suite (Release 7)

Blackboard Learning System™

Blackboard Community System™

Blackboard Content System™

Data Integration Manual (Doc #170013) Publication Date: October 13, 2005

WORLDWIDE HEADQUARTERS	INTERNATIONAL HEADQUARTERS
Blackboard Inc.	**Blackboard International B.V.**
1899 L Street, NW, 5th Floor Washington, DC 20036-3861 USA	Keizersgracht 62/64 1015 CS Amsterdam The Netherlands
800-424-9299 toll free US & Canada	
+1-202-463-4860 telephone	+31 20 5206884 (NL) telephone
+1-202-463-4863 facsimile	+31 20 5206885 (NL) facsimile
www.blackboard.com	global.blackboard.com

Blackboard, the Blackboard logo, Blackboard Academic Suite, Blackboard Learning System, Blackboard Learning System ML, Blackboard Community System, Blackboard Transaction Suite, Building Blocks, and Bringing Education Online are either registered trademarks or trademarks of Blackboard Inc. in the United States and/or other countries. Intel and Pentium are registered trademarks of Intel Corporation. Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States and/or other countries. Sun, Solaris, UltraSPARC, and Java are either registered trademarks or trademarks of Sun Microsystems, Inc. in the United States and/or other countries. Oracle is a registered trademark of Oracle Corporation in the United States and/or other countries. Red Hat is a registered trademark of Red Hat, Inc. in the United States and/or other countries. Linux is a registered trademark of Linus Torvalds in the United States and/or other countries. Apache is a trademark of The Apache Software Foundation in the United States and/or other countries. Macromedia, Authorware and Shockwave are either registered trademarks or trademarks of Macromedia, Inc. in the United States and/or other countries. Real Player and Real Audio Movie are trademarks of RealNetworks in the United States and/or other countries. Adobe and Acrobat Reader are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. Macintosh and QuickTime are registered trademarks of Apple Computer, Inc. in the United States and/or other countries. WordPerfect is a registered trademark of Corel Corporation in the United States and/or other countries. Crystal Reports is a trademark of Crystal Decisions in the United States and/or other countries. WebEQ is a trademark of Design Science, Inc. in the United States and/or other countries. JSpell is a trademark of The Solution Café in the United States and/or other countries. Other product and company names mentioned herein may be the trademarks of their respective owners. Patents pending.

© 2005 Blackboard Inc. All rights reserved. Made and printed in the USA.

No part of the contents of this manual may be reproduced or transmitted in any form or by any means without the written permission of the publisher, Blackboard Inc.

Table Of Contents

© 2005 Blackboard Inc. Proprietary and Confidential

© 2005 Blackboard Inc. Proprietary and Confidential

INTRODUCTION

Overview

The *Blackboard Academic Suite Advanced Integration and Data Management Manual* provides background information on integration, as well as specific instructions and resources for using the integration tools included with *Blackboard Academic Suite* ™ (Release 7.0).

Keep in mind that, while the integration tools are only included with the *Blackboard Learning System*, the tools can be used to integrate *Blackboard Community System* Organization data with other systems for clients who have this license. These tools are also available to clients with only the *Blackboard Community System*. The text of this document references the *Blackboard Learning System* when referring to both Course and Organization data; however, Organizations are only enabled with the licensing of the *Blackboard Community System*.

Data integration

Data integration is the efficient and timely automatic updating of data from an institution system to the *Blackboard Academic Suite*. To create this automated process, planning and implementation must occur. Blackboard Technical Solutions and Blackboard Technical Support provide the necessary support to institutions required for successful data integration. Blackboard Technical Support may be contacted at http://behind.blackboard.com/

In this manual

The Blackboard Academic Suite Advanced Integration and Data Management Manual contains the following sections:

- Principles of Data Integration
- Data Integration Tools
- Appendix - Code Samples
- Appendix - Data Format Tables
- Glossary of Terms

Please note that the *Blackboard Academic Suite Advanced Integration and Data Management Manual* is updated periodically. Check the Date Last Update at the beginning of the manual to ensure that it is the most recent copy. Any updates are listed in the Appendix.

To report any comments or suggestions regarding this manual, please contact Blackboard Support.

© 2005 Blackboard Inc. Proprietary and Confidential

PRINCIPLES OF DATA INTEGRATION

Overview

This section reviews the relationship between institutional data and Blackboard data, the data integration process, and the methods of integrating institutional data with *Blackboard Academic Suite*.

What is data integration?

Data integration involves the transfer of user, course, enrollment, and staff data between an institution's information systems and Blackboard systems. The information model used by the *Blackboard Academic Suite* is that defined by the Instructional Management System (IMS) consortium. IMS is a global effort whose primary objective is to define a standardized set of structures that can be used to exchange data between different instruction-based systems. IMS has defined an information model and provided a specification for transfer of such data.

For more information, please visit the IMS Web site: http://www.imsproject.org/.

In this section

This section contains the following topics:

© 2005 Blackboard Inc. Proprietary and Confidential

DATA ENTITIES

Overview

A data entity is a type or group of data records that share the same attributes. An easy way to visualize a data entity is to view it as a table, with attributes as columns and records as rows. Each record in an entity is defined by a unique identifier called a primary key, or a key. The *Blackboard Academic Suite* adheres to the IMS standards, making it possible to consistently map data entities from a variety of sources to data entities in the Blackboard platform.

Data entities

Data that can be input to the *Blackboard Academic Suite* are grouped into the following ten entities:

- **User**: Data consists of user role, ID, contact, and personal information.
- **Course**: Data consists of course name, course section ID, and description information.
- **Organization**: Data consists of organization name, organization section ID, and description information.
- **Student enrollment**: Data consists of the core information required to assign a Student to a given course section.
- **Staff assignment**: Data consists of the core information required to assign a staff member to a given course section.
- **Organization membership**: Data consists of the core information required to establish a user's membership in a given organization.
- **Course Category**: Data consists of the title, key, and availability information of a course catalog category.
- **Course Category Membership**: Data consists of the information required to connect a course to a category.
- **Organization Category**: Data consists of the title, key, and availability information of an organization catalog category.
- **Organization Category Membership**: Data consists of the information required to connect an organization to a category.

Blackboard Academic Suite data entities

The table below details the *Blackboard Academic Suite* data entities that have a corresponding IMS data entity.

BLACKBOARD DATA ENTITY	IMS DATA ENTITY	DESCRIPTION
User	Person	An end user of the *Blackboard Academic Suite*, including students, staff, and others.

© 2005 Blackboard Inc. Proprietary and Confidential

BLACKBOARD DATA ENTITY	IMS DATA ENTITY	DESCRIPTION
Course	Group	An instance of an online class delivered through the *Blackboard Learning System*, typically a course section.
Organization	Group	A site for groups, organizations, clubs, and teams to have an online presence, share information, engage in increased communication, and collaborate on tasks.
Student enrollment	Group membership	The record establishing a student's participation in a Course Web site.
Staff assignment	Group membership	The record establishing a staff member's participation in a course Web site and their role in the delivery of the course.
Organization membership	Group membership	The record establishing a user's membership in an organization and their role in the organization.

© 2005 Blackboard Inc. Proprietary and Confidential

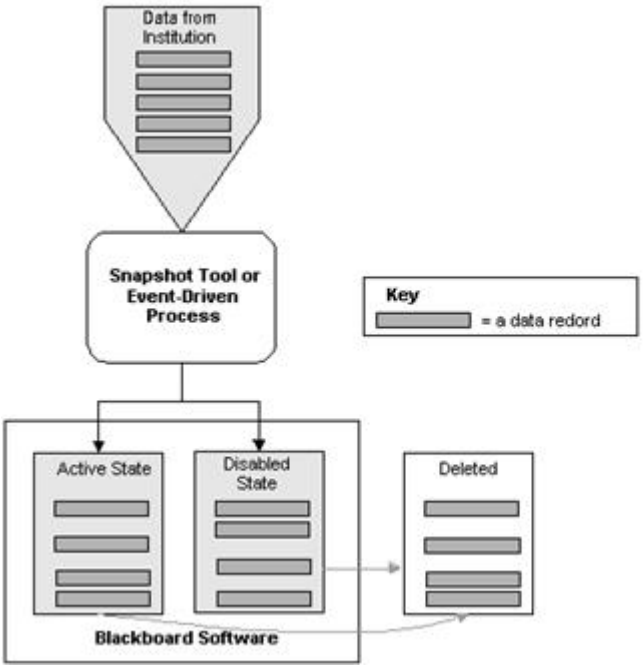
DATA LIFECYCLE

Overview

The data lifecycle is the series of events that control the actions of a record from creation to when it is deleted. Disabling a record rather than deleting it is often the best option if the record will be activated at a later time. A data record in the Blackboard platform has a lifecycle of usefulness and associated status that it can pass through as its usefulness changes. The system processes the data differently based on its state in the lifecycle. Please note that courses and organizations are the only data entities that may be archived through the user interface and introduced into the system at a later date through the interface.

Data lifecycle

Each data record has a lifecycle. A data lifecycle is the creation, update, and archiving or deletion of a record over time.



Triggering events

A triggering event is an event that impacts (adds to, updates, or deletes) data within the Blackboard platform. When planning data integration, written definitions of triggering events and complete documentation of the criteria for adding, updating, disabling, or deleting data will increase the long-term integrity of the system and promote organizational consistency.

Data states

A record with a status of active can be disabled or deleted. A disabled object can be activated or deleted. Disabling active objects and enabling inactive objects is done through the Snapshot Tool.

© 2005 Blackboard Inc. Proprietary and Confidential

Deleting a record can be accomplished through the integration tools or through the System Control Panel user interface. Course and organization content are the only data entities that can be archived before being deleted. Archived data can be retrieved and inserted into the Blackboard platform through the user interface.

Behavior of data records in relation to their state

The table below details each entity and a description of its records' behavior in certain states.

ENTITY	STATE	BEHAVIOR IN BLACKBOARD
User	Active	Default state. The user record appears in the system.
	Disabled	The user record exists within the database but is not active within the system. The user cannot login however the user record appears in the System Control Panel in gray text with an icon that signals its disabled status.
	Deleted	Deletes user data and course and organization membership.
Course or Organization	Active	Default state. The course appears within the *Blackboard Learning System*.
	Disabled	The course does not appear to any users within the *Blackboard Learning System* user interface, however, it still exists in the database. The record appears in the System Control Panel in gray text with an icon that signals its disabled status.
	Deleted	Deletes all course content, enrollments, and instructor assignments and all activity data associated with the course.
Student Enrollment or Organization Membership	Active	Default state. The enrollment appears within the *Blackboard Learning System*.
	Disabled	The student enrollment record is in the database but cannot be viewed through the user interface except by the Administrator. The record appears in the System Control Panel in gray text with an icon that signals its disabled status.
	Deleted	The enrollment record is deleted from *Blackboard Learning System* and all activity associated with the enrollment with the exception of Discussion Board messages.
Instructor Assignment	Active	Default state. The assignment appears within the *Blackboard Learning System*.
	Disabled	The Assignment record appears in the database but not in the user interface. The Instructor or staff member may not access the course.

© 2005 Blackboard Inc. Proprietary and Confidential

ENTITY	STATE	BEHAVIOR IN BLACKBOARD
	Deleted	The record is deleted from *Blackboard Learning System*.
Course or Organization Category	Active	Default state. The category appears within the *Blackboard Learning System*.
	Disabled	The category exists in the database but does not appear in the *Blackboard Learning System*. Courses listed under the category are not deleted.
	Deleted	The category does not exist in *Blackboard Learning System*.
Course or Organization Category Membership	Active	Default state. The link appears within the *Blackboard Learning System*.
	Disabled	The link exists in the database but does not appear in the *Blackboard Learning System*.
	Deleted	The link does not exist in *Blackboard Learning System*.

Behavior of records in active state

The table below details the behavior of active entities within *Blackboard Academic Suite* based on availability.

ENTITY	AVAILABILITY	BEHAVIOR IN *BLACKBOARD ACADEMIC SUITE*
User	Available	Default. Can access features and functions of Blackboard as defined by role.
	Unavailable	User cannot log in. Only System Administrators can see the record. The User Name appears as gray on the user interface. Does not automatically disable membership in a course or organization.
Course or Organization	Available	Default. Can be accessed by all authorized users.
	Unavailable	Only System Administrators and the course Instructor can view and access the course. Does not automatically remove enrollments and Instructor assignments.
Student Enrollment or Organization Membership	Available	Default. Enables users to have access to their courses.
	Unavailable	Instructors and System Administrators can see the unavailable student enrollment within the course membership list. The record appears as gray on the user interface.
Instructor Assignment	Available	Default. Enables Instructors to access their courses.

© 2005 Blackboard Inc. Proprietary and Confidential

ENTITY	AVAILABILITY	BEHAVIOR IN *BLACKBOARD ACADEMIC SUITE*
	Unavailable	Instructors and System Administrators can see the unavailable Assignment within the course membership list. The record appears in gray and the course is inaccessible to the Instructor.
Course or Organization Category	Available	Default. The category appears within the *Blackboard Learning System* and courses may be assigned to it.
	Unavailable	The category appears as gray text in the *Blackboard Learning System*. While it is visible, courses may not be assigned to it.
Course or Organization Category Membership	Available	Default. Courses linked to a category appear under that category.
	Unavailable	A list of courses attached to an unavailable category will appear to the Administrator, but the category will appear in gray text.

© 2005 Blackboard Inc. Proprietary and Confidential

INTEGRATION PROCESS

Overview

The integration process is a collaborative effort between Blackboard Academic Suite and the institution. Blackboard Technical Support and Blackboard Technical Services provide the support, experience, and knowledge necessary to assist institutions with an efficient integration.

Questions to ask prior to integration

Institutions must answer the following questions prior to data integration.

- Which institutional systems contain the data for integration with the *Blackboard Academic Suite*?
- How will attributes of the institutional system map to the *Blackboard Academic Suite*?
- Which criteria must be satisfied before a data record can be sent from the institutional system to the *Blackboard Academic Suite*?
- How will integration provide the needed packaging and error handling practices required by the institution?

Project teams

To successfully plan and implement a data integration project, Blackboard recommends forming cross-functional teams representing each stakeholder in the project. These teams usually consist of:

- an institutional Project Manager
- institutional analysts with detailed understanding of the institution's business processes, needs, and associated information systems
- institutional programmers with the ability to query, report from, and interface with the enterprise systems

If the institution is working with Blackboard Technical Solutions on the data integration project, these teams should also represent Blackboard, the institution, and solution partners. These teams may consist of:

- an institutional Project Manager
- a Blackboard Project Manager
- institutional analysts with detailed understanding of the institution's business processes, needs, and associated information systems
- institutional programmers with the ability to query, report from, and interface with the enterprise systems

© 2005 Blackboard Inc. Proprietary and Confidential

- Blackboard and solution-partner technical consultants

Steps to a successful integration

The following includes the seven steps to a successful integration.

1. Define the scope and time frame of the data integration project: what data entities will be exchanged between the systems and when will the project be complete?
2. Develop, analyze, and document requirements, including ownership of data entities and attributes, key definition, attribute mapping, data modeling and trigger determination, timeliness, reliability, security, and operational issues.
3. Design, configure, and program a data-integration solution: although *Blackboard Learning System* and *Blackboard Community System* features many built-in integration capabilities, some institutions may require additional customization; where appropriate, the project team can use existing solutions from enterprise vendors, reuse prior solutions developed for other institutions, or develop an entirely new custom solution.
4. Develop and document operating procedures for all affected parties.
5. Test the system.
6. Transfer the technology to the operations team at the institution.
7. Train all relevant personnel.

Data integration planning process

Steps two and three in the table above, document the planning, design, configuration and programming of the data integration process. The planning portion consists of developing the following:

- **Data mapping**: organize the data sources to prepare the data dictionary, and map required and optional attributes of *Blackboard Learning System* to those of institution systems.
- **Data lifecycle**: generate business rules that regulate how data is handled.
- **Data operations**: generate rules that govern the use of the integration tools to transfer data.

For assistance in planning data integration, please contact Blackboard Technical Services.

Tips and Tricks

- Complete the data integration for one information system at a time; it's quicker to add a second auxiliary system after the first is done.
- When planning the integration, focus on getting the data into the system before determining when to remove it; in most cases, institutions will have a whole semester or longer to complete the rules for disabling data so the plan to remove it does not have to be completed up front.

© 2005 Blackboard Inc. Proprietary and Confidential

DATA MAPPING

Overview

To integrate institutional systems data with the *Blackboard Learning System*, institutional data attributes must be mapped to attributes in the *Blackboard Learning System* data dictionary. The Blackboard Data Dictionary appears in the Reference Materials section of this manual.

Blackboard Learning System attributes

Blackboard Learning System has a minimal set of required attributes (fields) for records of each entity type. This minimal set of attributes provides the basic integration capability. By using additional attributes available in *Blackboard Learning System*, more complex integration can be performed for a richer and more convenient online learning experience.

Ownership of information

Only authorized personnel should have access to the systems to perform changes. Designating such data as owned by the institutional system will override any changes to that data made through the *Blackboard Learning System* user interface. For example, if a user were to change their password through the *Blackboard Learning System* interface, the next time data was transferred from the institutional system, the password attribute in *Blackboard Learning System* would be overwritten if that attribute where owned by the institutional system. In this example, the institution system can control the password field without conflict if the user ability to change their password is disabled within *Blackboard Learning System*.

NOTE Institution systems will not contain data corresponding to each attribute in *Blackboard Learning System*. Data that appears only in *Blackboard Learning System* should not be owned by the institution system.

Data in multiple enterprise systems

For data records that appear in multiple institution systems, a decision must be made as to which key takes precedence during data integration.

© 2005 Blackboard Inc. Proprietary and Confidential

DATA LIFECYCLE PLANNING

Overview

Data lifecycle planning defines the rules that control how the institution manages data. These rules will determine what data is transferred to *Blackboard Learning System*, when it is transferred, as well as any other conditions for data handling.

Data lifecycle planning

Data lifecycle planning involves the creation of business rules. Business rules are statements that define or constrain some aspect of the business. They assert structure to control or influence the behavior of the business. Business rules cannot be broken down into smaller pieces. The business rules set by an institution will control whether or not data may be created or changed, when data can be sent, and the implications when this occurs.

Important business rules

Some important business rules to consider are:

- Where data are stored?
- What business processes impact the data?
- Which records to send?
- How often to send records?
- When to stop sending specific records?
- When to archive course Web site and organization Web site records?

© 2005 Blackboard Inc. Proprietary and Confidential

DATA OPERATIONS PLANNING

Overview

Data Operations planning involves choosing the data integration tool or tools that best suits the needs of the institution and applying the business rules developed during the data lifecycle planning.

Importing data into the Blackboard Learning System database

Blackboard Learning System imports administrative data via a data feed, which can be done using a flat file, an XML file, or the Event Driven API (Application Program Interface). A data feed request can be:

- Add a new record
- Update an existing record
- Delete an existing record
- Change the key of a record

Warning: Single server conversions are not supported. Changing the key or unique identifier, of a record should be done sparingly and with the greatest care. However, there are some situations where it is necessary. For example, if a Social Security number is used as the key for user records. Foreign students are assigned a temporary Social Security number, when they enroll this will be used as the key. When a foreign student receives a permanent Social Security number, it makes sense to change the key within the system. This means that clients may not install Release 7.0 on the same server as Blackboard Learning System – ML.

Methods of integration

There are two basic methods of data integration: Using the Event Driven APIs and Snapshot. The two methods can be combined into a composite method, configured according to institution specifications, that uses the strengths of each method while off-setting each method's limitations.

Event Driven APIs approach

The Event Driven APIs allow institutional systems and the *Blackboard Learning System* database to exchange event information. When an event occurs (add, update, or delete) that changes the state of the data in the institution systems, a transaction is sent to the *Blackboard Learning System* database to provide synchronization between the two systems.

Under the Data Integration approach, when a triggering event takes place, a command is automatically sent to the *Blackboard Learning System* database. The *Blackboard Learning System* database receives the command, executes it, and returns a status update to the information system. If a failure occurs, the command is saved or the event is "marked" until a backup takes place and the command can be re-sent successfully. To implement this approach, access to triggers in the enterprise system is required.

© 2005 Blackboard Inc. Proprietary and Confidential

Snapshot-based approach

Snapshot allows all data for an entity to be transferred from an institutional information system to the *Blackboard Learning System* database at a particular point in time. It is called a snapshot because it is akin to taking a picture of the institutional database at that point in time and transferring the contents to the *Blackboard Learning System* database. In a snapshot-based approach, snapshots are taken at pre-determined times by the Snapshot Generator. The Snapshot Generator then creates a snapshot file from the relevant information and the Snapshot Tool transfers that file to the *Blackboard Learning System* database.

The Snapshot-based approach offers higher failure tolerance than the Data Integration approach, but reduces synchronization. Snapshot is best used to handle large batches of data. Using this approach, reports of relevant data ("snapshots") are run at regularly scheduled times. The Snapshot Tool sends commands to the *Blackboard Learning System* database instructing it to update any data that has changed since the last comparison. In the event of a *Blackboard Learning System* database failure, Administrators can run another snapshot or process the last snapshot. Because the entire data set must be compared each time the Snapshot Tool is run, this approach's performance is slower than the Data Integration approach. Dividing snapshots into logical subsets, using data source keys, can alleviate some of the problems associated with a bulky Snapshot process.

Data sources allow for:

- smaller files
- faster processing
- different update patterns

Please see the topic on the Data Source Management Tool (DSM) for more information on constructing data sets.

Initial population of the *Blackboard Learning System* database is easily accomplished using the snapshot-based solution.

Combined Snapshot/ Event Driven API approach

The combined approach attempts to use the best aspects of both methods. First, the Snapshot approach is used to achieve improved failure tolerance at the expense of performance. Then, the important events that require quick synchronization with the *Blackboard Learning System* database are handled using the Event Driven API approach. Using this approach, an institution can reduce the complexity of managing failures while maintaining a high level of synchronization. It can be implemented for all triggering events or just the most significant.

© 2005 Blackboard Inc. Proprietary and Confidential

DATA INTEGRATION TOOLS

Overview

The *Blackboard Learning System* includes tools and APIs that help institutions integrate *Blackboard Learning System* with their information systems. The Integration tools allow programmers, database administrators, and network engineers to develop software that transfers data from the institution, including student information systems (SIS), to Blackboard.

In this section

This section of the *Blackboard Learning System Advanced Integration and Data Management Manual* includes detailed information on the Snapshot Tool as well as the Event Driven API. The information is presented to advanced programmers and administrators with a minimum of introductory material.

Snapshot Tool

The Snapshot Tool offers administrators a command line interface to update the *Blackboard Learning System* database with information contained in a flat file or XML file. The *Blackboard Learning System* database can be updated with a manually controlled explicit operation (Manual mode) or through Snapshot mode. Snapshot mode automatically compares data in a flat file to the *Blackboard Learning System* database and performs logic based on the results of the comparison. The data in the *Blackboard Learning System* database will be updated or disabled based upon the logical comparison performed by the Snapshot Tool.

Event Driven API

The Event Driven API provides a collection of Java classes that programmers can use to insert, update, delete, or disable information in the *Blackboard Learning System* database based on data extracted from an institution's information systems. While similar to the Snapshot Tool in that each achieves data integration, the Event Driven API allows an institution to create robust software to manage the data link. Event Driven APIs are commonly used to update data in the *Blackboard Learning System* database based on an external event that triggers the data transaction. The update is made in real or near-real time; the *Blackboard Learning System* database is updated when the new information is entered into the institution system.

Often, Web-based systems use these APIs to create special workflows. Some examples include:

- guest management
- organization creation and management
- payments for enrollments

Contents

This section of the Blackboard Learning System Advanced Integration and Data Management Manual contains the following topic:

- Integration Tools Installation for clients

© 2005 Blackboard Inc. Proprietary and Confidential

Snapshot Method

- Snapshot Process
- Snapshot Components
- Snapshot Files
- Snapshot File Formats
- Connectivity Properties
- Data Source Management Tool*
- Sample Naming Convention for Data Source Keys
- Command Line Syntax
- Operation
- Data Feed Properties

Event Driven API

- Event Driven API Architecture
- Data Integration Process
- Data Package
- Persister Package

*Also used with the Event Driven API

INSTALL INTEGRATION TOOLS ON A CLIENT MACHINE

Installing the client software

Follow the steps below to install the *Blackboard Learning System* client tools. The client tools can process the Snapshot files instead of the application server, leaving more of the application server resources available to handle other tasks. The most recent version of JDK 1.3.1 and Perl are required to run the client tool.

Instructions for Windows Operating System:

1. Download the Snapshot Client Installer.
2. From the directory the file was unpacked into, run the appropriate Blackboard6Client_windows file.
3. Read and accept the License Agreement.
4. Click **Browse** to locate and select a License file.
5. Select the directory where the client tools will be installed. If left blank, a Blackboard directory will be created.
6. Enter the location of the JDK.
7. Enter the database server information, including, the database machine name, the database domain, SQL® Server Instance Name (if not using the default), the BbAdmin database user password and confirm the password.
8. The database server location information.

Instructions for UNIX Operating System:

1. Download the Snapshot Client Installer.
2. From the directory the file was unpacked into, run the appropriate file for the operating system. After downloading the file check that it is executable.

   ```
   Blackboard6Client_linux
   Blackboard6Client_solaris
   ```

3. Accept the license agreement.
4. Enter the full path to the license file including the name of the file.
5. Specify the directory that will hold the client tools.
6. Enter the path to the JDK directory.
7. Enter the database machine name, database domain name, Oracle SID, and bbadmin database user password.

NOTE: If the database machine is not resolved on the DNS server then it must be resolved in the /etc/hosts file of the local machine that will run the client tools.

Confirm that all the information entered is correct and proceed with the installation.

CLASSPATH

Implementations must include the following .jar files in the CLASSPATH to use the data integration tools. Also, any institutional system using the Event Driven API must also have the .jar files in the CLASSPATH.

Unix

/blackboard/systemlib/bb-platform.jar

/blackboard/systemlib/cms-admin.jar

/blackboard/systemlib/bb-snapshot.jar

/blackboard/systemlib/jdbc/Opta2000.jar

/blackboard/systemlib/jdbc/jdbc2_0-stdext.jar

/blackboard/systemlib/xerces-1.4.3.jar

/blackboard/systemlib/gnu-regexp-1.0.8.jar

/blackboard/systemlib/gnu-getopt-1.0.8.jar

/blackboard/systemlib/servelet.jar

/blackboard/systemlib/mail.jar

/blackboard/systemlib/activation.jar

/blackboard/systemlib/soap.jar

Windows

\blackboard\systemlib\bb-platform.jar

\blackboard\systemlib\cms-admin.jar

\blackboard\systemlib\bb-snapshot.jar

\blackboard\systemlib\jdbc\Opta2000.jar

\blackboard\systemlib\jdbc\jdbc2_0-stdext.jar

\blackboard\systemlib\xerces-1.4.3.jar

\blackboard\systemlib\gnu-regexp-1.0.8.jar

\blackboard\systemlib\gnu-getopt-1.0.8.jar

\blackboard\systemlib\servelet.jar

\blackboard\systemlib\mail.jar

\blackboard\systemlib\activation.jar

\blackboard\systemlib\soap.jar

The required .jar files are added to the **CLASSPATH** during installation. In most cases, they only need to be manually added to the **CLASSPATH** of other institutional systems that will make use of the Event Driven API. Web services must be restarted after changing the **CLASSPATH**.

© 2005 Blackboard Inc. Proprietary and Confidential

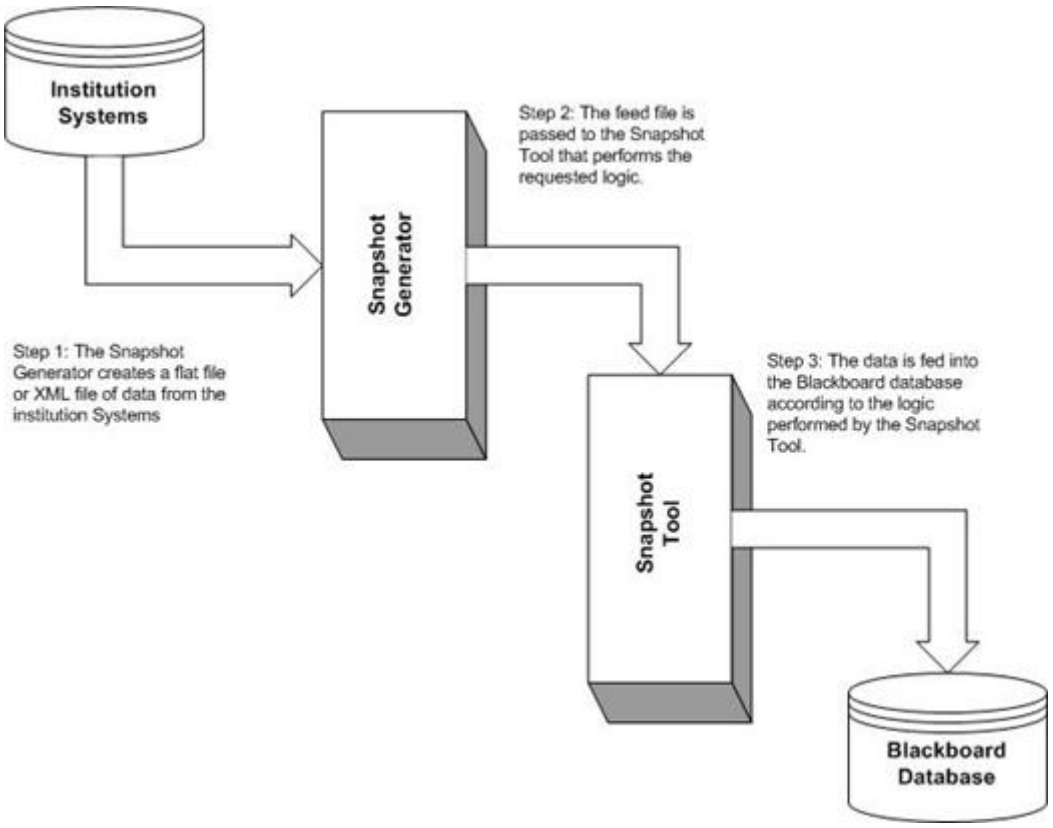
SNAPSHOT METHOD

Overview

The Snapshot Tool offers Administrators a command line interface to update the *Blackboard Learning System* database with information contained in a flat file or XML file. This topic reviews the Snapshot process. Each of the components of a Snapshot feed is described in the following topics.

Process diagram

The diagram below depicts how data is transferred from institution systems to *Blackboard Learning System* using the Snapshot process.



Frequency of Snapshot data feeds

Data can be loaded into the *Blackboard Learning System* database on a schedule determined by the institution and defined by the institution's business rules. The frequency of data feeds can change throughout the year for different data entities.

© 2005 Blackboard Inc. Proprietary and Confidential

The following are some of the factors that should be considered when determining the frequency of data feeds:

- Frequency of adding, modifying, and deleting data in institution systems
- Urgency of the data synchronization
- Institution system performance
- Number of data entities to be fed from institution systems into *Blackboard Learning System*

To set the frequency, use the Task Scheduler with Windows NT or Windows 2000 operating systems and the **cron** command with Linux® or Solaris® operating systems. In addition, Windows 2000 users may also use the **at** Scheduler or the **at** command from the command prompt. Windows NT users can use **winat** available from the Resource Kit or the **at** command at the DOS prompt.

NOTE: Frequency can also be controlled with the Snapshot Controller. The Snapshot Controller may be built by the institution or provided through Blackboard Technical Solutions.

SNAPSHOT BASICS

Overview

The Snapshot Tool offers Administrators a command line interface to update the Blackboard database with information contained in a flat file or XML file. This section provides a brief description of the Snapshot process and some examples. All the tools necessary to perform the snapshot and Data Source Management (DSM) commands are automatically installed as part of the *Blackboard Learning System*.

Data Source Management (DSM) command

This command shows the data source keys that accompany the Snapshot Tool. A Data Source Key is created to provide a means of separation between entities. The `readme.txt` file contains a list of all the available commands and can be used as a reference.

Snapshot command

The Administrator uses a Snapshot Generator to extract data from the institution's database and insert it into a flat file or XML file. A Snapshot Generator is a tool written in JAVA or another language that uses SQL commands for data manipulation.

The Snapshot command is used to insert, update or purge data, such as user, courses, and instructors, into or out of the *Blackboard Learning System* database. Snapshot commands are often run by the Administrator directly from an application server.

Examples

There are many instances in which an institution may benefit from using Snapshot. The following are a few examples, with a brief description of the process used to execute them.

NOTE: In the following examples, "webserver host" indicates that users should insert the name of their webserver host.

Example 1

Organize Students in the Information Systems Management program into two groups; Project Management (PM) and ISD (Information System Development).

1. Create two data source keys; IFSM.PM.02 and IFSM.ISD.02
2. Associate the data source key to the selected user group in the `snapshot.properties` file. Insert the users with the desired Snapshot command:

   Designates flag to enable course sub group copying for configuration property file `groups.copy=Y`

   Designates flag to enable membership data copying for configuration property file `membership.copy=Y`

© 2005 Blackboard Inc. Proprietary and Confidential

Windows

```
C:\blackboard\apps\snapshot\bin>snapshot –V “webserver host” -f
usr_snpsht –t ..\Demo\users\ISD.02.csv -C
..\data\snpshot.properties
```

UNIX

```
/usr/local/blackboard/snapshot/bin >./snapshot.sh –V “webserver
host” -f usr_snpsht –t ../Demo/users/ISD.02.csv -C
../data/snpshot.properties
```

3. Run the same process again using snapshot command with the file PM.02.csv associated with the key IFSM.PM.02

Example 2

A Student is disabled in the system until their financial aid arrives. Disabling a record rather then deleting it is often the best option if the record will be activated at a later time. When a user is disabled the user record exists within the database but is not active in the *Blackboard Learning System*. The user cannot login. However, the user record appears in the System Control Panel in gray text with an icon that signals its disabled status.

A Student can be disabled by changing an element in the student record and running the snapshot command again using the same source key.

Example 3

Organize a Statistics course so that it is offered in two sections over the Summer 2002 semester.

1. Create data source keys for the course with two sections, offered in a Summer 2002 semester.

   Windows

   ```
   C:\blackboard\apps\snapshot\bin\dsm –V “webserver host” –f create
   –b stats.101.SUM.10 –d “Statistics 101 – section 10 summer 2002”

   C:\blackboard\apps\snapshot\bin\dsm –V “webserver host” –f create
   –b stats.101.SUM.11 –d “Statistics 101 – section 11 summer 2002”
   ```

   UNIX

   ```
   /usr/local/blackboard/apps/snapshot/bin/dsm.sh –V “webserver host”
   –f create –b stats.101.SUM.10 –d “Statistics 101 – section 10
   summer 2002”

   /usr/local/blackboard/apps/snapshot/bin/dsm.sh –V “webserver host”
   –f create –b stats.101.SUM.11 –d “Statistics 101 – section 11
   summer 2002”
   ```

2. Set the properties to the desired key each time a course is inserted
3. Insert the first and second sections of the Statistics course.

   Windows

   ```
   C:\blackboard\apps\snapshot\bin>snapshot –V “webserver host” -f
   crs_snpsht =t ..\demo\courses\stat_sun_02_10.csv –C
   ..\data\snapshot.properties

   C:\blackboard\apps\snapshot\bin>snapshot –V “webserver host” -f
   crs_snpsht =t ..\demo\courses\stat_sun_02_11.csv –C
   ..\data\snapshot.properties
   ```

© 2005 Blackboard Inc. Proprietary and Confidential

UNIX

```
/usr/local/blackboard/apps/snapshot/bin >./snapshot.sh –V
“webserver host” -f crs_snpsht =t
../demo/courses/stat_sun_02_10.csv –C ../data/snapshot.properties

/usr/local/blackboard/apps/snapshot/bin >./snapshot.sh –V
“webserver host” -f crs_snpsht =t
../demo/courses/stat_sun_02_11.csv –C ../data/snapshot.properties
```

Example 4

Enroll Students in a course. The file used to enroll students into a course uses the same idea as a relational database. The file to enroll users is created by combining external_person_key and external_course_key

1. Create a key for the course enrollment

   Windows

   ```
   C:\blackboard\apps\snapshot\bin>dsm –V “webserver host” –f create
   –b enr.stat.10.sum –d “enrollment for Statistics 101 section 10
   for summer of 2002 semester”
   ```

   UNIX

   ```
   /usr/local/blackboard/apps/snapshot/bin/dsm.sh –V “webserver host”
   –f create –b enr.stat.10.sum –d “enrollment for Statistics 101
   section 10 for summer of 2002 semester”
   ```

2. Set this key in the `snapshot.properties` file before running the snapshot command.
3. Enroll selected Students into the course.

© 2005 Blackboard Inc. Proprietary and Confidential

SNAPSHOT COMPONENTS

Overview

This topic describes each of the logical components that typically comprise the Snapshot method. Please see the Snapshot Process topic for information on how all the components fit together to integrate the *Blackboard Learning System* with institution systems.

Snapshot Generator

The Snapshot Generator is software that is written by the institution. The Generator extracts data from institution systems, one file per type per datasource, and creates flat files or XML files that are then fed into *Blackboard Learning System*.

Snapshot Tool

The Snapshot Tool is a simple, command line program that takes a feed file and pushes the data into the *Blackboard Learning System* database. It is built for bulk operations and has several modes that can be used to facilitate different workflows.

Snapshot Controller

The Snapshot Controller helps run the Snapshot Tool for each snapshot file generated. The controller automates and manages the process from the time a data feed leaves the Snapshot Generator until it reaches the *Blackboard Learning System* database. The Snapshot Controller software can be customized for your institution to direct email to appropriate people if a failure occurs in the process. Thus, failure handling is relatively simple during the Snapshot process.

NOTE: The Snapshot Controller is not necessary for Data Integration. The Snapshot Controller is not included with the Blackboard Learning System and may be written by the institution. Please contact Blackboard Technical Solutions for assistance.

© 2005 Blackboard Inc. Proprietary and Confidential

SNAPSHOT FILES

Overview

Snapshot files can be character-delimited flat files or XML files that conform to IMS standards. Blackboard adheres to the Global Community XML encoding standards, which commonly uses UTF–8. This topic reviews in detail the format of Snapshot flat files and provides a link to information on IMS standards for XML files.

UNIX users: Snapshot files must be owned by bbuser when processed.

To accept multi byte information, files must be in Unicode format. All Unicode formatted files must include the Byte Order Mark (BOM). If a BOM is not specified, the system reads it as ISO-8859-1. The supported formats are as follows:

- UTF-8
- UTF-16LE
- UTF-16BE
- ISO-8859-1

ASCII is also still a supported format.

Snapshot XML files

An XML Snapshot file has increased flexibility, in that the XML files can push through one data type at a time or combine multiple data types to process at once. While *Blackboard Learning System* does not validate against the XML data dictionary (DTD) the code must be well-formed, meaning each nested tag must be properly closed.

The Snapshot Tool will process IMS compliant XML files, version 1.01. For information on IMS standards for XML files, view the following Web page on the IMS site:

http://www.imsproject.org/enterprise/enbind03.html

Since entities in the XML file are typed, there are no specific requirements to separate entities into separate feed files. However, please note that the action and data source attributes specified in the IMS standard are ignored. In processing the IMS XML format, the Blackboard Snapshot Tool still requires the same logic and workflow as used with a flat file.

The following refer to specific tags in the XML template:

- Extension tags: These tags are not supported in the IMS specifications; they are only supported by the *Blackboard Learning System*. As they are adopted by the IMS project, Blackboard will phase them into proper XML definition.
- Categories and Category Memberships: These are not IMS data types and are not supported by IMS; they are supported by Blackboard.

The following tips are helpful for troubleshooting Snapshot XML files:

- If a user fails to be created in an XML file then their membership will also fail. Many aspects of the XML file are related, therefore is one area fails, it may cause others to also fail.

© 2005 Blackboard Inc. Proprietary and Confidential

- Blackboard recommends using an XML viewer when viewing the .invalid file. This file should be referenced if an error occurs.

Snapshot flat files

The Snapshot flat file contains delimited data that are converted into native data types by the Snapshot tool and then fed into the *Blackboard Learning System* database. The default delimiter is a | (pipe). Clients may use any SQL character as the delimiter, just be aware it cannot appear un-escaped in the data. If there is no value for a field in a Snapshot record, that field is unchanged in *Blackboard Learning System* (Exception: if the password field is null the User ID will be set as the password). Use a single space as the value of a field in a Snapshot file to clear that field within *Blackboard Learning System*.

Some data feed elements have sets of string constants for their possible values. For example, the attribute Institution_Role maps to coded characters in the database.

Many times, an institution will keep track of information that does not match a data attribute in *Blackboard Learning System*. Any information included in the data file that does not match a *Blackboard Learning System* attribute will not be added to the *Blackboard Learning System*. This can be very useful for comments, forward planning, and similar tasks.

Invalid data

Incorrectly formatted data is handled as follows:

- Strings that exceed the specified max length for that column will be truncated. Keys that exceed the specified max length for that column will cause the record to fail.
- Data types that cannot be converted (such as dates) are ignored. (They are treated as a space or as missing.)
- Invalid column headers are ignored.

NOTE: Unsupported feed elements will not cause the feed to fail—they are ignored to preserve backward compatibility with older feed generators. It is anticipated, though, that many of these elements may be added in subsequent releases.

Snapshot error logs

The Snapshot Tool generates error log files. These files can accumulate over time and must be managed by the institution or through the Snapshot Controller.

Many *Blackboard Learning System* Administrators set up a **cron** command in UNIX or Linux environments or use the Task Scheduler or **at** command in Windows environments to remove log files periodically.

To manage error logs, use the Task Scheduler with Windows NT or Windows 2000 operating systems and the **cron** command with Linux or Solaris operating systems. In addition, Windows 2000 users may also use the **at** Scheduler or the **at** command from the command prompt. Windows NT users can use winat available from the Resource Kit or the **at** command at the DOS prompt.

General formatting rules

© 2005 Blackboard Inc. Proprietary and Confidential

Keep the following rules in mind when creating Snapshot flat files:

- A comma is the default column delimiter but the recommended delimiter format is a pipe "|". Please note that it is possible to select an escape character in the Snapshot.properties file.
- Records must be separated by a carriage return, a linefeed, or a combination of the two. This is not configurable.
- The file must begin with a header to identify the file's data field. The header will be on a single line listing the record fields included in the file. The header format will be delimited using the same character as the data records and include the exact names of the fields included in the file as listed in the data element tables in the appendix of this manual.
- Data files wishing to make use of a Snapshot Controller Tool to automate the system must include a footer that holds the characters "***FileFooter", a number equaling the number of data records included in the file, and a timestamp showing the time and date the file is completed in the format: HH:MM:SS MM/DD/YYYY. The footer need not be created manually; it is added to a file by the Snapshot Controller.
- The header and footer also provide points of reference to ensure that a Snapshot file is complete. If the file is created by the Snapshot Generator and either the header or the footer is missing, it is probable that all the required data is not included in the file.

© 2005 Blackboard Inc. Proprietary and Confidential

SNAPSHOT FILE FORMATS

Overview

There are three basic types of Snapshot files: the Snapshot feed file type, the delete records file type, and the manual update file. Each file format and examples are given below.

Snapshot feed file

The Snapshot feed file is the most commonly used file format and may encompass information on any of the ten entities that are transferred to the *Blackboard Learning System*. The file format for the Snapshot file is as follows:

Header: Field Names. The exact names of the fields included in the file. These names are given in the feed element tables in the Reference Materials section of this document.

Footer: The footer (created by the Snapshot Controller) holds the following information:

- ***FileFooter. Characters designating the end of the file.
- Record count: The number of data records included in the file.
- Timestamp: A system time stamp of the time and date the file is completed in the format: HH:MM:SS MM/DD/YYYY.

Example (Membership):

```
EXTERNAL_COURSE_KEY|EXTERNAL_PERSON_KEY|ROLE
Math101.1.Fall1999|1074202|course_builder
History176.6.Spring2000|324-765-0098|instructor
Chem401.1.Fall1999|uberk1278|student
***FileFooter|3|04:25:33 03/02/2001/
```

Note that only fields that contain data that is to be imported into *Blackboard Learning System* should be listed in the header and included in the subsequent data records. Information that is not given (left blank) for fields that are included in the header is not changed. To clear a field, enter a single space. Any fields not found in the Data Dictionary but added to the flat file will be ignored.

Delete records file

The `DeleteRecords` file will be the file format used to purge unwanted data from the *Blackboard Learning System* and may encompass information on any of the ten entities that are to be transferred to the *Blackboard Learning System*. Because only records that should be deleted are included in a `DeleteRecords` file, the file need only contain certain key fields in order to delete data records from *Blackboard Learning System*. For a list of the required fields for each entity type, please refer to the data dictionary tables in the Reference section of this manual. The file format for the `DeleteRecords` file is as follows:

- Header: Field Names. The exact names of the fields included in the file.
- Footer: Not Required. As this file type is designated as a manual process file, the footer used by the Controller software to automate the snapshot process is unnecessary.

© 2005 Blackboard Inc. Proprietary and Confidential

Example (Membership):

```
EXTERNAL_COURSE_KEY|EXTERNAL_PERSON_KEY
Math101.1.Fall1999|1074202|course_builder
History176.6.Spring2000|324-765-0098|instructor
Chem401.1.Fall1999|uberk1278|student
```

Manual update records file

The `ManualUpdateRecords` file will be the file format used to insert or update records but not disable records explicitly disabled in the Manual Update file. It may encompass information on any of the ten entities that are to be transferred to the *Blackboard Learning System*. The required fields for each entity type are the same as for the Snapshot files. The file format for the `ManualUpdateRecords` file is as follows:

- Header: Field Names. The exact names of the fields included in the file.
- Footer: Not Required. As this file type is designated as a manual process file, the footer used by the Controller software to automate the snapshot process is unnecessary.

Example (Membership):

```
EXTERNAL_COURSE_KEY|EXTERNAL_PERSON_KEY|ROLE
Math101.1.Fall1999|1074202|course_builder
History176.6.Spring2000|324-765-0098|instructor
Chem401.1.Fall1999|uberk1278|student
```

Note that only fields that contain data that is to be imported into *Blackboard Learning System* should be listed in the header and included in the subsequent data records. Information that is not given (left blank) for fields that are included in the header is not changed. To clear a field, enter a single space. Any fields not found in the Data Dictionary but added to the flat file will be ignored.

© 2005 Blackboard Inc. Proprietary and Confidential

XML FILE EXAMPLES

Below is a sample of an XML file:

```
<ENTERPRISE>
	<PROPERTIES lang="EN">
		<DATASOURCE>Blackboard University</DATASOURCE>
		<TARGET>Computing and Telecommunications LMS</TARGET>
		<TYPE>Snapshot</TYPE>
		<DATETIME>1999-12-21</DATETIME>
	</PROPERTIES>
	<PERSON>
		<SOURCEDID>
			<SOURCE>Blackboard University</SOURCE>
			<ID>39450210223
				<!-- Campus User Key -->
			</ID>
		</SOURCEDID>
		<USERID>swang</USERID>
		<NAME>
			<FN>Mr. Stanley Wang Jr.</FN>
			<SORT>Wang,Stanley</SORT>
			<NICKNAME>Wang,Stanley</NICKNAME>
			<N>
				<FAMILY>Wang
					<!-- Last Name -->
				</FAMILY>
				<GIVEN>Stanley
					<!-- First Name -->
				</GIVEN>
				<OTHER>Franklin
					<!-- Middle Name -->
				</OTHER>
				<PREFIX>Mr.
					<!-- Title -->
				</PREFIX>
				<SUFFIX>Jr.</SUFFIX>
			</N>
```

© 2005 Blackboard Inc. Proprietary and Confidential

```
                </NAME>
                <DEMOGRAPHICS>
                        <GENDER>2</GENDER>
                        <BDAY>1959-01-01
                                <!-- Birthday -->
                        </BDAY>
                </DEMOGRAPHICS>
                <EMAIL>Swang5@Blackboard_university.com
                        <!-- User Email -->
                </EMAIL>
                <TEL teltype="1">3104591276
                        <!-- Telphone. 0-Home Phone 1, 1-Home Fax, 2-
WorkPhone 1, 3-Work Fax, 4-Mobile Phone, 5-Home Phone 2, 6-WorkPhone 2 --
>
                </TEL>
                <TEL teltype="2">3104591200
                        <!-- Telphone. 0-Home Phone 1, 1-Home Fax, 2-
WorkPhone 1, 3-Work Fax, 4-Mobile Phone, 5-Home Phone 2, 6-WorkPhone 2 --
>
                </TEL>
                <ADR>
                        <STREET>Twin Oaks Valley Rd
                                <!-- Address Line 1 -->
                        </STREET>
                        <STREET>attn: S. Wang
                                <!-- Address Line 2 -->
                        </STREET>
                        <LOCALITY>San Marcos
                                <!-- City -->
                        </LOCALITY>
                        <REGION>CA
                                <!-- State Province -->
                        </REGION>
                        <PCODE>92096-0001
                                <!-- Zip postal code -->
                        </PCODE>
                        <COUNTRY>US
                                <!-- Country -->
                        </COUNTRY>
                </ADR>
                <DATASOURCE>Blackboard University</DATASOURCE>
```

© 2005 Blackboard Inc. Proprietary and Confidential

```
                <EXTENSION>
                        <X_BB_SYSTEMROLE>0
                                <!-- Campus Role. 0-System Admin, 1-System
Support, 2-Course Creator, 3-Account Admin, 4-User, 5-None -->
                        </X_BB_SYSTEMROLE>
                        <X_BB_INSTITUTIONROLE>0
                                <!-- Instution role. 0-Student, 1-Faculty, 2-
Staff, 3-Alumni, 4-Prospective Student, 5-Guest, 6-Other, 7-
Administratior -->
                        </X_BB_INSTITUTIONROLE>
                        <X_BB_STUDENTID>144532
                                <!-- Person Id -->
                        </X_BB_STUDENTID>
                        <X_BB_PASSWORD>rpeterson
                                <!-- User Password -->
                        </X_BB_PASSWORD>
                </EXTENSION>
        </PERSON>
        <PERSON>
                <SOURCEDID>
                        <SOURCE>Blackboard University</SOURCE>
                        <ID>12345678910
                                <!-- Campus User Key -->
                        </ID>
                </SOURCEDID>
                <USERID>jinstructor</USERID>
                <NAME>
                        <FN>John Instructor</FN>
                </NAME>
                <DEMOGRAPHICS>
                        <GENDER>2</GENDER>
                        <BDAY>1949-03-01
                                <!-- Birthday -->
                        </BDAY>
                </DEMOGRAPHICS>
                <EMAIL>Swang5@Blackboard_university.com
                        <!-- User Email -->
                </EMAIL>
                <DATASOURCE>Blackboard University</DATASOURCE>
        </PERSON>
        <GROUP transaction="1">&#10;
```

```
                <SOURCEDID>&#10;
                        <SOURCE>UW-Superior</SOURCE>&#10;
                        <ID>Summer2001_MATH_112_W06</ID>&#10;
                </SOURCEDID>&#10;
                <DESCRIPTION>&#10;
                        <SHORT>Summer2001_MATH_112_W06</SHORT>&#10;
                        <LONG>Intro To Contemporary Math</LONG>&#10;
                        <FULL>The questions &quot;What is mathematics used
for?
 What do mathematicians do? What do they practice or&#10;believein?&quot;
are raised and answered. Emphasis on developing the capacity to think
logically and critically&#10;and to develop strong conceptual
understanding and appreciation rather than computational expertise.
Topics&#10;are selected from management science, statistics, social
choice, and computer science. Students see how&#10;mathematics is used to
solve current problems.Satisfies the Mathematics requirement for General
Education.</FULL>&#10;
                </DESCRIPTION>&#10;
                <ORG>&#10;
                        <ORGNAME>Dept of Math &amp; Comp
Science</ORGNAME>&#10;
                </ORG>&#10;
                <TIMEFRAME>&#10;
                        <BEGIN restrict="0">2001/06/11</BEGIN>&#10;
                        <END restrict="0">2001/08/22</END>&#10;
                        <ADMINPERIOD>Summer2001</ADMINPERIOD>&#10;
                        </TIMEFRAME>&#10;
                        <ENROLLCONTROL>&#10;
                                <ENROLLACCEPT>1</ENROLLACCEPT>&#10;
                        </ENROLLCONTROL>&#10;
                <DATASOURCE>Extended Degree</DATASOURCE>&#10;
        </GROUP>
        <GROUP>
                <SOURCEDID>
                        <SOURCE>Blackboard University</SOURCE>
                        <ID>0390(1999-Fall)COMPSCI_697C_Sec._1-1164
                                <!-- Course Section Key -->
                        </ID>
                </SOURCEDID>
                <DESCRIPTION>
                        <SHORT>Security_In_Computing
                                <!-- Abbreviated Title -->
                        </SHORT>
```

© 2005 Blackboard Inc. Proprietary and Confidential

```
                    <LONG>Graduate Level Special Topics course covering
security in computing today.
                          <!-- Title -->
                    </LONG>
                    <FULL>This course will examine threats and security
issues in today's common computing environments. Prerequisites: Advanced
Networks (CS 622) and Cryptography (CS 633).
                          <!-- Course Description -->
                    </FULL>
             </DESCRIPTION>
             <ORG>
                    <ORGNAM>Computer Science</ORGNAM>
             </ORG>
             <TIMEFRAME>
                    <BEGIN restrict="0">1999-09-01
                          <!-- Start Date -->
                    </BEGIN>
                    <END restrict="0">1999-12-15
                          <!-- End Date -->
                    </END>
                    <ADMINPERIOD>Fall 1999</ADMINPERIOD>
             </TIMEFRAME>
             <ENROLLCONTROL>
                    <ENROLLACCEPT>1</ENROLLACCEPT>
                    <URL>www.psunv.class.f1999</URL>
             </ENROLLCONTROL>
             <DATASOURCE>Blackboard University</DATASOURCE>
             <EXTENSION>
                    <BKBHIERARCHY>PSUNV LBART UGRD 0390 MATH10
1164</BKBHIERARCHY>
                    <X_BB_GROUP_TYPE>0
                          <!-- Service Level. 0-Course, 1-Organization -
->
                    </X_BB_GROUP_TYPE>
                    <X_BB_TEMPLATEKEY>src101
                    </X_BB_TEMPLATEKEY>
             </EXTENSION>
      </GROUP>
      <MEMBERSHIP>
             <SOURCEDID>
                    <SOURCE>Blackboard University</SOURCE>
                    <ID>0390(1999-Fall)COMPSCI_697C_Sec._1-1164
```

```
                        <!-- Course Section Key -->
                    </ID>
                </SOURCEDID>
                <MEMBER>
                    <SOURCEDID>
                        <SOURCE>Blackboard University</SOURCE>
                        <ID>39450210223
                            <!-- Campus user Key -->
                        </ID>
                    </SOURCEDID>
                    <IDTYPE idtype="1"/>
                    <ROLE roletype="01">
                        <!-- Course Role. 0-Instructor, 1-Teaching
Assistant, 2-Course Builder, 3-Grader, 4-Student, 5-Guest, 6-
None                    -->
                        <STATUS>1</STATUS>
                        <COMMENTS>Enrolled</COMMENTS>
                        <DATE>1999-09-01</DATE>
                        <TIMEFRAME>
                            <BEGIN restrict="0">1999-09-01</BEGIN>
                            <END restrict="0">1999-12-15</END>
                        </TIMEFRAME>
                        <FINALRESULT>
                            <MODE>Graded</MODE>
                            <RESULT>B</RESULT>
                        </FINALRESULT>
                    </ROLE>
                </MEMBER>
                <MEMBER>
                    <SOURCEDID>
                        <SOURCE>Blackboard University</SOURCE>
                        <ID>12345678910
                            <!-- Campus user Key -->
                        </ID>
                    </SOURCEDID>
                    <IDTYPE idtype="1"/>
                    <ROLE roletype="02">
                        <!-- Course Role. 0-Instructor, 1-Teaching
Assistant, 2-Course Builder, 3-Grader, 4-Student, 5-Guest, 6-
None                    -->
                        <TIMEFRAME>
```

```
                        <BEGIN restrict="0">1999-09-01</BEGIN>
                        <END restrict="0">1999-12-15</END>
                    </TIMEFRAME>
                    <SUBROLE>Primary Instructor</SUBROLE>
                    <STATUS>1</STATUS>
                </ROLE>
            </MEMBER>
            <DATASOURCE>Blackboard University</DATASOURCE>
        </MEMBERSHIP>
</ENTERPRISE>
```

Category and Category Membership

The following is an example of the Category and CategoryMembership data types:

```
<ENTERPRISE>
        <CATEGORY type="0">
            <SOURCEDID>
                <SOURCE>Blackboard University</SOURCE>
                <ID>crs_cat
                    <!-- Course Category Key -->
                </ID>
            </SOURCEDID>
            <EXTENSION>
                <X_BB_TITLE>
                    Course Category
                </X_BB_TITLE>
                <X_BB_AVAILABLE >
                    Y
                </X_BB_AVAILABLE >
            </EXTENSION>
        </CATEGORY>
        <CATEGORY type="1">
            <SOURCEDID>
                <SOURCE>Blackboard University</SOURCE>
                <ID>org_cat
                    <!-- Organization Category Key -->
                </ID>
            </SOURCEDID>
            <EXTENSION>
```

© 2005 Blackboard Inc. Proprietary and Confidential

```
                <X_BB_TITLE>
                      Organization Category
                </X_BB_TITLE>
                <X_BB_AVAILABLE >
                      Y
                </X_BB_AVAILABLE>
          </EXTENSION>
    </CATEGORY>
    <CATEGORY_MEMBERSHIP type="0">
          <SOURCEDID>
                <SOURCE>Blackboard University</SOURCE>
                <ID>0390(1999-Fall)COMPSCI_697C_Sec._1-1164
                      <!-- Course Section Key -->
                </ID>
          </SOURCEDID>
          <CATEGORY_MEMBER>
                <SOURCEDID>
                      <SOURCE>Blackboard University</SOURCE>
                       <ID>crs_cat
                            <!-- Course Category Key -->
                      </ID>
                </SOURCEDID>
          </CATEGORY_MEMBER>
    </CATEGORY_MEMBERSHIP>
    <CATEGORY_MEMBERSHIP type="1">
          <SOURCEDID>
                <SOURCE>Blackboard University</SOURCE>
                <ID>0390(1999-Fall)COMPSCI_697C_Sec._1-1164
                       <!-- Course Section Key -->
                </ID>
          </SOURCEDID>
          <CATEGORY_MEMBER>
                <SOURCEDID>
                      <SOURCE>Blackboard University</SOURCE>
                      <ID>org_cat
                            <!-- Organization Category Key -->
                      </ID>
                </SOURCEDID>
          </CATEGORY_MEMBER>
    </CATEGORY_MEMBERSHIP>
```

© 2005 Blackboard Inc. Proprietary and Confidential

```
</ENTERPRISE>
```

© 2005 Blackboard Inc. Proprietary and Confidential

CONFIGURATION OF THE SERVICE

Overview

The configuration properties, stored in configuration files, determine the Snapshot Tool operating parameters. It is common to all invocations of the Snapshot Tool. Under most conditions it is not necessary to make changes to the properties files as the values are set at installation.

Please note that it is necessary to invoke a properties file.

Properties files

The configuration properties files are stored in the /**blackboard/config** directory (blackboard\config for Windows users). With Snapshot and the Event Driven APIs, the following properties files are used:

service-config-snapshot-soap.properties

service-config-snapshot-jdbc.properties

These files are configured at install. If a change needs to be made, backup these files and use them as templates for new config files that include the desired properties.

NOTE: All properties files are formatted similar to Windows.ini files, with name/value pairs. Comment lines begin with a hash mark (#).

Invoke property file for Data Integration

It is necessary to invoke a properties file when running the Event Driven APIs from a client machine. The file That should be invoked is:

service-config-snapshot-soap.properties

Database configuration file

The following two files must be changed if the Administrator changes the bbadmin password:

/blackboard/config/bb-datastores.xml

/blackboard/config/bb-datastores-standalone.xml

Initialize persistence with SOAP

Simple Object Access Protocol (SOAP) is a set of conventions used for invoking code using XML over HTTP. When programming using the Event Driven APIs, the SOAP password (MD5 encrypted) must be passed when run from a client. For example:

```
System.getProperties().setProperty(CIConstants.REMOTE_PASSWORD,"password"
);
```

```
java "-Dremote.access.password=password"
```

© 2005 Blackboard Inc. Proprietary and Confidential

Set the context for Virtual Installations

The context must be set when running against a Virtual Installation.

© 2005 Blackboard Inc. Proprietary and Confidential

DATA SOURCE MANAGEMENT TOOL

Overview

The Data Source Management (DSM) tool is used to identify and manage data coming from external data sets. All data that is pushed to the *Blackboard Learning System* database is tagged with a data source identifier. A data source is a logical set of data, sometimes simply associated with a source system, other times associated with type and term data as well. This allows a flexible way to manage data from multiple external sources into a single *Blackboard Learning System* database, or to identify all data for a given semester. For example, an institution may choose to identify all Fall 2000 records with the data source key SIS.FALL2000.

Please note that the Data Source Management tool can be used with Snapshot and the Event Driven APIs.

Configuration properties

The Data Source Management Utility uses the same configuration file as the Snapshot Tool.

Listing and creating data sources

The ability to list and create data sources is handled directly by the Data Source Management tool. Those commands provided in *Blackboard Learning System* for listing and creating datasources (`Listdatasources.sh` for UNIX or `Listdatasources.cmd` for Windows and `createdatasource.sh` for UNIX and `createdatasource.sh` for Windows) no longer function in *Blackboard Learning System*.

NOTE: It is important to separate like data entities into different data sources. For example, including courses and organizations in the same data set will process the entire data set according to the applied logic and will not differentiate between a course and an organization.

Data source keys

A data source key is an attribute used to identify a set of data. The key is used to isolate data that will be affected during Snapshot operations. A data source key must be identified before any Snapshot operations can be performed. The sets of data are typically associated with the system that originated the data, such as an institution's student information system.

There is no meaning assigned to the key or its parts other than as an identifier. For administrative purposes, though, it is useful to adhere to a naming convention for the keys to assign some meaning, and thus be able to easily distinguish between data sets based on the key. This is because there are many ways to group the data that is going to be sent to the *Blackboard Learning System* database. Please note that the Snapshot Controller does assign meaning to the key.

© 2005 Blackboard Inc. Proprietary and Confidential

NOTE: When assigning data sources to course and organization categories, child categories must belong to the same data source as the parent category when the category tree is inserted. If child categories do not appear in the same data source as the parent, the child-parent relationship will not be maintained.

Command line operations

The following operations can be performed at the command line.

OPERATION	DESCRIPTION
LIST	To list all datasources.
CREATE	To create a new datasource.
REMOVE	To remove a datasource.
MODIFY	To modify a datasource. The modify command can be used to change either the description, the key, or both.
COUNTS_ALL	To list all entity counts for a datasource. The count includes both enabled and disabled records.
COUNTS_DISABLED	To list disabled entity counts for a datasource.
COUNTS_ENABLED	To list enabled entity counts for a datasource.
DISABLE_ALL	To disable all entities for a datasource.
DISABLE_PERSON	To disable all person entities for a datasource.
DISABLE_GROUP	To disable all group entities for a datasource.
DISABLE_MEMBERSHIP	To disable all membership entities for a datasource.
DISABLE_COURSE_CATEGORIES	To disable all course category entities for a datasource. Note that all category memberships related to this category are not disabled in the database but are rendered unusable due to the missing category.

© 2005 Blackboard Inc. Proprietary and Confidential

OPERATION	DESCRIPTION
DISABLE_ORGANIZATION_CATEGORIES	To disable all organization category entities for a datasource. Note that all category memberships related to this category are not disabled in the database but are rendered unusable due to the missing category.
DISABLE_COURSE_CATEGORY_MEMBERSHIP	To disable all course category memberships for a datasource.
DISABLE_ORGANIZATION_CATEGORY_MEMBERSHIP	To disable all organization category memberships for a datasource.
PURGE_ALL	To purge all entities for a datasource. Purge only deletes disabled items.
PURGE_PERSON	To purge all person entities for a datasource. Purge only deletes disabled items.
PURGE_GROUP	To purge all group entities for a datasource. Purge only deletes disabled items.
PURGE_MEMBERSHIP	To purge all membership entities for a datasource. Purge only deletes disabled items.
PURGE_COURSE_CATEGORIES	To purge all course category entities for a datasource. This will also purge all category memberships related to the category. Purge only deletes disabled items.
PURGE_ORGANIZATION_CATEGORIES	To purge all organization category entities for a datasource. This will also purge all category memberships related to the category. Purge only deletes disabled items.

© 2005 Blackboard Inc. Proprietary and Confidential

OPERATION	DESCRIPTION
PURGE_COURSE_CATEGORY_MEMBERSHIP	To purge all course category memberships for a datasource. Purge only deletes disabled items.
PURGE_ORGANIZATION_CATEGORY_MEMBERSHIP	To purge all organization category memberships for a datasource. Purge only deletes disabled items.
PURGE_SESSION	To clear all persistent storage devices used by snapshot.

Command line syntax

To use the DSM operations, use the appropriate script for the operation as listed in the table below. The **key** argument is the value that the Snapshot Tool will use to identify records in a given data set. The **description** argument is simply a human-readable description of the data source. Allowed characters are letters, numbers, and spaces. If including spaces, encase the description argument in quotation marks.

NOTE: To use the following invocations, the Virtual host identification must be added as shown:

```
script –V “webserver host”
```

If running the invocation from a remote location (SOAP), the integration user account password information must be added as shown:

```
script –V “webserver host” –P “password”
```

The “integration” user account is used for the authentication of SOAP requests.

OPERATION	SCRIPT
LIST	-f *operation*
CREATE	-f *operation* –b *key* –d *description*
REMOVE	-f *operation* –b *key*
MODIFY	-f *operation* –b *key* –d *description* –r *replacement key*
COUNTS_ALL	-f *operation* –b *key*
COUNTS_DISABLED	-f *operation* –b *key*
COUNTS_ENABLED	-f *operation* –b *key*
DISABLE_ALL	-f *operation* –b *key* –t *date*

© 2005 Blackboard Inc. Proprietary and Confidential

OPERATION	SCRIPT
DISABLE_PERSON	-f *operation* –b *key* –t *date*
DISABLE_GROUP	-f *operation* –b *key* –t *date*
DISABLE_MEMBERSHIP	-f *operation* –b *key* –t *date*
DISABLE_COURSE_CATEGORIES	-f *operation* –b *key* –t *date*
DISABLE_ORGANIZATION_CATEGORIES	-f *operation* –b *key* –t *date*
DISABLE_COURSE_CATEGORY_MEMBERSHIP	-f *operation* –b *key* –t *date*
DISABLE_ORGANIZATION_CATEGORY_MEMBERSHIP	-f *operation* –b *key* –t *date*
PURGE_ALL	-f *operation* –b *key* –t *date*
PURGE_PERSON	-f *operation* –b *key* –t *date*
PURGE_GROUP	-f *operation* –b *key* –t *date*
PURGE_MEMBERSHIP	-f *operation* –b *key* –t *date*
PURGE_COURSE_CATEGORIES	-f *operation* –b *key* –t *date*
PURGE_ORGANIZATION_CATEGORIES	-f *operation* –b *key* –t *date*
PURGE_COURSE_CATEGORY_MEMBERSHIP	-f *operation* –b *key* –t *date*
PURGE_ORGANIZATION_CATEGORY_MEMBERSHIP	-f *operation* –b *key* –t *date*
PURGE_SESSION	-f *operation*
DISABLE_INSTITUTION_ROLE_MEMBERSHIP	To disable all Institution Roles for a datasource.
PURGE_INSTITUTION_ROLE_MEMBERSHIP	To purge all Institution Roles entities for a datasource. Purge only deletes disabled items.

*optional. (Date format is always yyyy/mm/dd)

NOTE: Disabled entities can be enabled through the Snapshot Tool in either Snapshot or Manual Update mode.

SAMPLE NAMING CONVENTION FOR DATA SOURCE KEYS

Overview

The following naming convention represents a relatively simple way to break up the data sets to enable the two most common workflows.

Data source ID

A simple ID should be assigned for the source system, for example SIS for a student information system, or HRMS for a human resources management system. By combining this ID with an ID for each type of set, a flexible naming scheme can be derived to support typical workflows.

Type bound sets

Type bound sets include a component that is derived from the type of feed that is being performed. This way a data set is identified by the entities involved. For example, if the string 'COURSE' is used to mean "Course" then that string is included to indicate the type of the data set, for example SIS.COURSE.

Term bound sets

Term bound sets are used to group data that are related, but should not overlap time periods in the database. For example, it might be desirable to feed spring courses into the database while fall courses are still active. Using a key that distinguishes the two sets based on their term will prevent snapshot operations on one set from interfering with data in the other. For example, SIS.SPRING2000 and SIS.FALL2000.

Type and term bound sets

In many cases it is desirable to use a combination of type and term bound identification. The most common example is Student enrollment at an institution with a fixed academic calendar—enrollment is bound to a specific semester for example, SIS.COURSE.FALL2000.

Example

An institution wants to process Student rosters, Instructor lists, course section listings, and enrollments over the course of several semesters. In general, from semester to semester, the Student and Instructor lists will encompass the same basic data set. However the courses and enrollments will need to be processed on a per-semester basis. That is, from semester to semester, active students and staff will be treated as a single logical set (with fluid membership) while courses and enrollments will be treated as logically distinct sets that do not intersect from semester to semester.

One solution is to use type-bound keys for Students and Instructors, and type and term bound keys for courses and enrollment. A data source key is created called SIS.USERS that is used to identify the set of users over time. By running Snapshot feeds on all ongoing users, all active Students, and staff may be processed as a single set of data.

© 2005 Blackboard Inc. Proprietary and Confidential

Separate data source keys are created for courses, enrollment, and Instructor Assignments, all which are both type and term bound:

SIS.COURSE.FALL2000

SIS.COURSE.SPRING2000

That way, all user feeds may use the **SIS.USERS** data source key, while courses and enrollment can use the SIS.COURSE.* keys, with guarantees that operations performed on one of the sets will not affect the other set.

As another example, different sets can be applied to different users:

MEDSIS.USERS

SIS.USERS

ALUMNI.USERS

© 2005 Blackboard Inc. Proprietary and Confidential

COMMAND LINE SYNTAX

Overview

This topic presents the command line syntax used to import a feed file into the *Blackboard Learning System* database. The following topics, Operations, COPYINTO Operations, and Data Feed Properties, review each variable of the command line syntax in detail.

Command line syntax

The following basic command line syntax should be run from the C:\blackboard\apps\snapshot\bin directory:

`snapshot –f operation –t datafile –C feed config file –V virtual hostname`

The **`datafile`** and **`feed config file`** variables must be the exact paths to those files.

Script files follow a naming syntax based on the operating system. For Windows installations, they are followed by the suffix "**`.cmd`**" and on UNIX installations they are followed by the suffix "**`.sh`**". For example, a Windows file would be **`snapshot.cmd`** and a UNIX file would be **`snapshot.sh`**.

Logs from the Snapshot operation are stored in the snapshot folder located at: Blackboard/logs/snapshot folder.

The file names include:

- for JDBC: bb-services-log-snapshot-jdbc.txt
- ϕor SOAP: bb-services-log-snapshot-soap.txt

After the Snapshot Tool is run the output of the Status Report includes a hex string. The following is an example:

"Status Report. --Process complete-- {6A6B3B403C5811D78248293C4666A3F3 }"

This hex string is a unique identifier generated by the Snapshot invocation. This unique identifier is used by Product Support for locating results when searching a detailed log file and for database tasks.

Virtual Host Name

The virtual hostname variable **`(–V virtual hostname)`** in the syntax represents the Virtual Installation that the command should be run against. If there are no Virtual Installations, or if the command should be run against the original installation, simply use the host name of the Web/app server.

Data source override

To override the data source key, use the override command as shown in the example below (be sure to include the quotes).

snapshot_override "-Ddata.source.key=Key" -f CRS_SNPSHT

Overriding the data source key from the command line permits a change to the data source key for a single feed without editing the **`feed config file`**.

© 2005 Blackboard Inc. Proprietary and Confidential

Please note the example uses the default feed config file, **`snapshot.properties`**, so there is no need to specifically call a file.

Script files follow a naming syntax based on the operating system. For Windows installations, they are followed by the suffix "**`.cmd`**" and on UNIX installations they are followed by the suffix "**`.sh`**". For example, a Windows file would be **`snapshot_override.cmd`** and a UNIX file would be **`snapshot_override.sh`**.

© 2005 Blackboard Inc. Proprietary and Confidential

OPERATION

Overview

This topic reviews the different modes of operation for the Snapshot Tool and the command line syntax for selecting a mode of operation. The modes described in the Operations Value table on the next page are invoked with a specific entity, for example, -CRS_SNPSHT for “Course Snapshot”, ENR_SNPSHT for “Enrollment Snapshot”, and so forth.

Snapshot

The Snapshot Tool takes the snapshot file, an “image-in-time” of the source data, and performs logic to decide what to do for records in the Blackboard database:

- If the record is in the feed, but not in the *Blackboard Learning System* database, an insert is performed.
- If the record is in the feed and in the *Blackboard Learning System* database, an update is performed to the *Blackboard Learning System* database.
- If the record is not in the feed, but in the *Blackboard Learning System* database, the record is disabled. If a later Snapshot contains a record in the disabled state, the record is enabled and updated. The entity may only be re-enabled by including it in a subsequent snapshot file, not through the Blackboard user interface. For more information see the topic, Data Lifecycle.

Manual

Manual update is an explicit selection of records. The operation is a “smart update,” in which the entity is inserted if not present in the *Blackboard Learning System* database or updated if present. The manual update feature does not allow the disabling of a record. This means that users may use manual update to add a small number of records without worrying about the possibility of disabling records.

Remove

Purge data for each record identified in the feed file. This is a permanent delete. The record is permanently removed from the database, including any associations of the record (i.e. if the user record is removed, any enrollments and files associated with the user is permanently deleted. If a course is removed, any enrollments associated with the course are also deleted).

Update logic

When a record is updated, ownership of the data fields, if applicable, is processed so that data owned by *Blackboard Learning System* is not changed. Blank values in a data field are ignored and the data is not changed.

© 2005 Blackboard Inc. Proprietary and Confidential

Operation values

The table below lists the operation values available when entering a snapshot command. These values replace the *Operation* variable in the command line. The values below are case sensitive.

VALUE	DESCRIPTION
CRS_SNPSHT	Processes a snapshot feed of courses.
CRS_MANUAL	Performs a smart update of courses.
CRS_REMOVE	Removes the specified list of courses.
CRS_COPYINTO	Copies course content into an existing course.
ENR_SNPSHT	Processes a snapshot feed of enrollment. Please note that performing this operation on instructor or staff data will fail.
ENR_MANUAL	Performs a smart update of enrollment.
ENR_REMOVE	Removes the specified list of enrollments.
ORG_SNPSHT	Processes a snapshot feed of organizations.
ORG_MANUAL	Performs a smart update of organizations.
ORG_REMOVE	Removes the specified list of organizations.
ORG_COPYINTO	Copies organization data into an existing organization.
USR_SNPSHT	Processes a snapshot feed of users.
USR_MANUAL	Performs a smart update of users.
USR_REMOVE	Removes a feed of users.
STAFF_SNPSHT	Processes a snapshot feed of Instructors and Staff.
STAFF_MANUAL	Performs a smart update of Instructors and Staff.
STAFF_REMOVE	Removes all Instructors and Staff in the feed file.
ORGMEM_SNPSHT	Processes a snapshot for organization membership. Please note that performing this operation on Instructors, organization Managers, or Staff data will convert those Instructors, Managers, or Staff to members within the organization.
ORGMEM_MANUAL	Performs a smart update for organization membership.
ORGMEM_REMOVE	Removes all organization memberships in the feed file.
CRS_CATEGORY_SNPSHT	To process a snapshot feed of course categories.
CRS_CATEGORY_MANUAL	To process a manual feed of course categories.

© 2005 Blackboard Inc. Proprietary and Confidential

VALUE	DESCRIPTION
CRS_CATEGORY_REMOVE	Removes feed of course categories.
CRS_CATEGORY_MEM_SNPSHT	To process a snapshot feed of Course Category Memberships.
CRS_ CATEGORY_MEM _MANUAL	To process a manual feed of Course Category Memberships.
CRS_ CATEGORY_MEM _REMOVE	Removes feed of Course Category Memberships.
ORG_CATEGORY_SNPSHT	To process a snapshot feed of organization categories.
ORG_CATEGORY_MANUAL	To process a manual feed of organization categories.
ORG_CATEGORY_REMOVE	Removes feed of organization categories.
ORG_ CATEGORY_MEM _SNPSHT	To process a snapshot feed of Organization Category Memberships.
ORG_ CATEGORY_MEM _MANUAL	To process a manual feed of Organization Category Memberships.
ORG_ CATEGORY_MEM _REMOVE	Removes feed of Organization Category Memberships.
INSTITUTION_ROLE_MEM_MANUAL	Performs a smart update for Institution Roles.
INSTITUTION_ROLE_MEM_REMOVE	Removes all Institution Roles in the feed file.
INSTITUTION_ROLE_MEM_SNPSHT	To process a snapshot feed of Institution Roles.

XML files

The table below lists the valid operation commands for handling XML files.

VALUE	DESCRIPTION
XML_SNPSHT	Performs a snapshot on an IMS-compliant XML file.
XML_MANUAL	Performs a smart update on an IMS-compliant XML file.
XML_REMOVE	Removes the entities specified in the IMS-compliant file.
XML_COPYINTO	Processes an XML copyinto feed.

© 2005 Blackboard Inc. Proprietary and Confidential

COPYINTO OPERATIONS

Overview

COPYINTO operations are a powerful way to manage course and organization content. The commands can be used to create templates that add the same content to many different courses or organizations. The COPYINTO command can be issued as part of a Snapshot or Manual operation, or as a stand-alone operation by invoking the **CRS_COPY_INTO** or **ORG_COPY_INTO** command.

COPYINTO behavior

When a course or organization is referenced in the **Template_Course_Key** or **Template_Organization_Key**, the COPYINTO operation will process the information according to the table below. Please note that in instances where a copy occurs on update, if the information from the source has already been added to the destination course, it will be added again resulting in duplicate data.

OPERATION	OWNERSHIP OF TEMPLATE KEY	RESULT
COPYINTO update	Blackboard	Copy
	not Blackboard	Copy
COPYINTO insert	Blackboard	Copy
	not Blackboard	Copy
Snapshot update	Blackboard	No Copy
	not Blackboard	Copy
Snapshot insert	Blackboard	Copy
	not Blackboard	Copy
Manual update	Blackboard	No Copy
	not Blackboard	Copy
Manual insert	Blackboard	Copy
	not Blackboard	Copy

Data feed properties

The specific components of the source Course or Organization to be copied can be configured in the data feed properties, described in the Data Feed Properties topic.

© 2005 Blackboard Inc. Proprietary and Confidential

© 2005 Blackboard Inc. Proprietary and Confidential

ASSIGN INSTITUTION ROLES THROUGH SNAPSHOT

Overview

Institution Roles are an expanded feature of Application Pack 2. Please see the *Blackboard Academic Suite Administrator Manual* for more information about Institution Roles. Institution Roles can be attached to users through Snapshot and the APIs. For information on using the APIs to assign Institution Roles please see the Administrative API specifications.

Snapshot Files

Snapshot User files include the INSTITUTION_ROLE element. This represents the Primary Institution Role and is expressed as the Role ID. Secondary Institution Roles are added through a new Snapshot file type, User Institution Role Membership. Each record matches one User to one Institution Role. To match a user to multiple Institution Roles requires multiple records in the file. The table below outlines the fields included in a record in a Snapshot User Institution Role Membership file.

ELEMENT	DESCRIPTION
EXTERNAL_PERSON_KEY	A unique identifier for a user at the institution This ID is provided by the institution and is not displayed to users. String. Max length 64. Multi byte characters accepted. Not Null, External Key
ROLE_ID	Institution Role. Passed as a string. Not null. Multi byte characters accepted.

Commands

The following command line operations handle User Institution Role Membership files.

OPERATION	DESCRIPTION
INSTITUTION_ROLE_MEM_MANUAL	Performs a smart update for Institution Roles.
INSTITUTION_ROLE_MEM_REMOVE	Removes all Institution Roles in the feed file.

© 2005 Blackboard Inc. Proprietary and Confidential

OPERATION	DESCRIPTION
INSTITUTION_ROLE_MEM_SNPSHT	To process a snapshot feed of Institution Roles.

DSM Parameters

The following operations can be used with the DSM tool.

OPERATION	DESCRIPTION
DISABLE_INSTITUTION_ROLE_MEMBERSHIP	To disable all Institution Roles for a datasource.
PURGE_INSTITUTION_ROLE_MEMBERSHIP	To purge all Institution Roles entities for a datasource. Purge only deletes disabled items.

© 2005 Blackboard Inc. Proprietary and Confidential

DATA FEED PROPERTIES

Overview

This topic reviews the properties that are contained within a data feed properties file. This file is represented by the –FEEDCFG parameter in the command line instructions. A data feed properties file determines how the Snapshot Tool will interpret the records in the data file. It is used on a per-feed basis.

NOTE: All properties files are formatted similar to Windows.ini files, with name/value pairs. Comment lines begin with a hash mark (#).

Data ownership

If the *Blackboard Learning System* database is declared as the owner of an attribute, then a difference between the feed file and the *Blackboard Learning System* database is ignored during a snapshot operation. If an institution system is declared the owner of an attribute, then values in the *Blackboard Learning System* database are changed to match the institution system during a snapshot operation. Note that data ownership is processed only during an update.

Data feed properties

This is the file specified by the **`feed config file`** parameter. This will most likely need to be edited before running a Snapshot operation. A default file is included as **`snapshot.properties`**.

NOTE: Do not use a Blackboard feed properties file with *Blackboard Learning System*.

Property	Description
max.error.count	Maximum number of errors allowed before processing of the current file terminates. Records that have been successfully processed are not rolled back. A setting of "-1" will allow an infinite number of errors during processing. A setting of "0" will terminate the process at the first error.
Data.delimiter	The column delimiter for the input file. This field is ignored for XML-based feeds. Setting the data delimiter allows for other delimiters to appear in the data. For example, if a comma is used in the data, select another delimiter to prevent errors. By default, the delimiter is a pipe "|".

© 2005 Blackboard Inc. Proprietary and Confidential

Property	Description
parse.allow.default	Enables the System Administrator to allow incorrect data in the snapshoot feed to be set to the default. “Y” or “N”. Default is “N”
entity.bb.controlled.fields	Attributes that are owned by the Blackboard Learning System database. That is, they should not be overwritten by the manual update or snapshot. On insert, the initial value in the feed will be entered into the database. However, on update, the value will not be changed. This is an effective way of updating user information without overwriting user passwords, fax numbers, or other similar information. Entity (as a variable in the property is) is one of person, group, membership, category, or categorymembership. The value of this property is a comma-delimited list of columns (attributes).
encrypt.password	Flag enabling automatic MD5 encryption of passwords provided. “Y” or “N”. The default is “Y”
encrypt.card.number	Flag enabling the system to accept encrypted card numbers. "N" signifies that the card number is not encrypted. "Y" signifies that the card number is already encrypted. Card numbers are unecrypted using a key shared between the Blackboard Transaction System and the Blackboard Learning System. If the card number is not encrypted at import, it will be encrypted by the Blackboard Learning System using the shared key. Therefore, it is very important that this flag is set correctly, otherwise, card numbers will be processed incorrectly.
escape.character	Character to be used in escaping delimiter in use. The default is “/”.
log.stdout	Flag handling display of status information to console. “Y” or “N” The default value is “N”.
assessment.copy	Flag that enables assessments for COPYINTO operations. “Y” or “N” If this is set to “N”, assessments that appear in a source course or organization will not be copied into the destination course. Default value is “Y”.
chat.archive.copy	Flag that enablesAnnouncements for COPYINTO operations. “Y” or “N” If this is set to “N”, announcements that appear in a source course or organization will not be copied into the destination course. Default value is “Y”.
chat.session.copy	Flag that enables chat sessions for COPYINTO operations. “Y” or “N” If this is set to “N”, sessions that appear in a source course or organization will not be copied into the destination course. Default value is “Y”.

© 2005 Blackboard Inc. Proprietary and Confidential

Property	Description
discussion.board.archive.copy	Flag that enables Discussion Board archives for COPYINTO operations. “Y” or “N” If this is set to “N”, archives that appear in a source course or organization will not be copied into the destination course. Default value is “Y”.
drop.box.copy	Flag that enables drop box items for COPYINTO operations. “Y” or “N” If this is set to “N”, drop box items that appear in a source course or organization will not be copied into the destination course. Default value is “Y”.
tasks.copy	Flag that enables tasks for COPYINTO operations. “Y” or “N” If this is set to “N”, tasks that appear in a source course or organization will not be copied into the destination course. Default value is “Y”.
announcements.copy	Flag that enables Announcements for COPYINTO operations. “Y” or “N” If this is set to “N”, announcements that appear in a source course or organization will not be copied into the destination course. Default value is “Y”.
calendar.copy	Flag that enables calendar items for COPYINTO operations. “Y” or “N” If this is set to “N”, calendar items that appear in a source course or organization will not be copied into the destination course. Default value is “Y”.
categories.copy	Flag that enables course or organization Categories for COPYINTO operations. “Y” or “N” If this is set to “N”, Categories that appear in a source course or organization will not be copied into the destination course. Default value is “Y”.
staff.information.copy	Flag that enables Staff Information for COPYINTO operations. “Y” or “N” If this is set to “N”, staff information that appear in a source course or organization will not be copied into the destination course. Default value is “Y”.
discussion.board.copy	Flag that enables Discussion Boards for COPYINTO operations. “Y” or “N” If this is set to “N”, discussion boards that appear in a source course or organization will not be copied into the destination course. Default value is “Y”.
course.content.copy	Flag that enables course content for COPYINTO operations. “Y” or “N” If this is set to “N”, course content that appears in a source course or organization will not be copied into the destination course. Default value is “Y”.
course.settings.copy	Flag that enables images and settings for COPYINTO operations. “Y” or “N” If this is set to “N”, images and settings that appear in a source course or organization will not be copied into the destination course. Default value is “Y”.

© 2005 Blackboard Inc. Proprietary and Confidential

Property	Description
groups.copy	Flag that enables course Group content for COPYINTO operations. “Y” or “N” If this is set to “N”, course group content that appears in a source course or organization will not be copied into the destination course. Default value is “Y”.
membership.copy	Flag that enables Student enrollments for COPYINTO operations. “Y” or “N” If this is set to “N”, Student enrollments that appear in a source course or organization will not be copied into the destination course. Default value is “Y”.
gradebook.copy	Flag that enables Gradebook content for COPYINTO operations. “Y” or “N” If this is set to “N”, Gradebook content that appears in a source course or organization will not be copied into the destination course. Default value is “Y”.
reconcile	Flag that, when set to “Y”, enables a Course Copy operation to completely replace the settings and Course Menu of the destination course. This works even when copying into a new course (removes the default settings and default Course Menu and replaces it with the source course). When the flag is set to “N” the source course will resolve itself against the destination course.
header.validation	Flag enabling validation of header information. Determines if labels provided are supported for current operation. “Y” or “N”
wait.length	Used to configure pause between persistence actions. (seconds) The default value is”-1”. where the wait length is not in use.
Snapshot.batch.size	Flag designating the number of records handled in a database transaction. Default is set to 300.
Error.delimiter	String used to mark the beginning of an error in the error log file. Default value is (!).
Data.source.key	External name of the data source used to identify entity sets from a given external source. The specified data source must first be defined with the DSM Utility before a snapshot or update is run with this key. This property can be overridden from the command line.
membership.datasource.key	Assigns a datasource key to the enrollments of a course that is created through Course Copy. If no datasource key is identified, the enrollments will use the same datasource key as the source.

© 2005 Blackboard Inc. Proprietary and Confidential

© 2005 Blackboard Inc. Proprietary and Confidential

ENABLING DATA INTEGRATION TOOLS TO USE SOAP

Overview

SOAP (Simple Object Access Protocol) is an XML format that handles the execution of server side code remotely, via the Web. Blackboard supports SOAP implementation of Loaders and Persisters in the Event Driven APIs. SOAP is activated through a change to the configuration. When this change is made, DSM and Snapshot will both use SOAP.

Configuration change

DSM and Snapshot share an environment file, which can be altered to use SOAP. This file is located at Blackboard/apps/snapshot/config/env.cmd.

By default the config file is configured directly to the database as:

```
CONFIG_FILE=%bbdir%\config\service-config-snapshot-jdbc.properties
```

To implement SOAP, change the property to read:

```
CONFIG_FILE=%bbdir%\config\service-config-snapshot-soap.properties
```

Command line syntax

When running the invocation through SOAP, the integration user account password information must be added as shown:

```
script –P “password”
```

Using an SSL Connection

Administrators may choose to use an SSL connection for SOAP Snapshots. The following instructions can be used for both Windows and UNIX systems.

1. **Locate the SSL certificate.** Check that the server SSL certificate is in X.509 form. If not, export the server’s SSL certificate to a file. On Windows, use the Certificate Export Wizard to export the certificate. On Unix, the certificate is located at the path set in

   `/usr/local/blackboard/apps/httpd/conf/ssl.conf`.

2. **Import the server SSL certificate to the cacerts keystore on the client.** The cacerts keystore is located in the Java JDK installation under jre/lib/security/cacerts. Use the keytool to import the server certificate, for example:

   ```
   keytool –import –file
   /usr/local/blackboard/apps/httpd/conf/certs/server.crt –keystore
   /usr/java/jdk1.3.1_07/jre/lib/security/cacerts –trustcacerts
   ```

   The Administrator will be prompted for a keystore password. The default keystore password for cacerts is “changeit”. The Administrator may change that password using the keytool.

   The server certificate may also be imported to a different keystore, but the Administrator

© 2005 Blackboard Inc. Proprietary and Confidential

should check the root Certificate Authority (CA) certificate from the issuing Certificate Authority exists in the cacerts keystore.

3. **Edit the service-config-snapshot-soap.properties file.** In this file, change the following key values:

```
blackboard.service.soap.param.encrypt=true
blackboard.service.soap.param.truststore=<location of keystore>
```

where <location of keystore> is replaced by the path to the keystore (see Step 2).

Additional information about the keytool utility may be found at the following Web sites:

Sun's instructions for this on Windows are found at:
http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html.

Sun's instructions for this on Unix are found at:
http://java.sun.com/j2se/1.3/docs/tooldocs/solaris/keytool.html.

© 2005 Blackboard Inc. Proprietary and Confidential

EVENT DRIVEN APIS

Overview

This section describes the *Blackboard Learning System* Event Driven API, which is used to programmatically push the following data into the *Blackboard Learning System* database from an external system:

- User
- Course
- Organization
- Enrollment
- Staff assignment
- Organization membership
- Course and organization category
- Course and organization category membership

The intended audience includes developers creating data integration solutions for the *Blackboard Learning System*. Code samples can be found in Appendix A – Code Samples.

NOTE: The Data Integration API Specification Guide describes the packages, classes, and objects of the Event Driven API in systemlib/cms-admin.jar

© 2005 Blackboard Inc. Proprietary and Confidential

EVENT DRIVEN API ARCHITECTURE

Overview

The API is a collection of java classes and objects that move data from the institution systems into the *Blackboard Learning System*. Concrete data from the institution systems are encapsulated as java objects. The methods contained in the java classes determine how the data are input into the *Blackboard Learning System*. The data input are controlled by persisters that process the appropriate method, convert the object into data that can be input into the *Blackboard Learning System* database, and then input that data according to the method called.

Entities and persisters

There are two main types of objects in the API: entities (objects) and persisters (actions). Entities include the objects that represent data in the system, such as users. Persisters are behind-the-scenes methods that handle the storage of the entities into a persistent store or transient data format.

Operations

All data classes have methods to handle persistence actions. The following persistence operations are supported:

- **Insert:** Inserts a record into the *Blackboard Learning System* database.
- **Update:** Updates an existing record in the *Blackboard Learning System* database.
- **Save:** Updates an existing record if it already exists. Otherwise, if it does not exist, inserts the record in the *Blackboard Learning System* database.
- **Remove.** Purges the record from the *Blackboard Learning System* database.
- **Change Key:** (Person and Group, Course, and Organization) Changes the primary key. This will automatically update any related Memberships of the changed keys.

Creating an object

To create an object in the system, instantiate a corresponding entity, set attributes on the object, and then call a persister method (insert, update, save, delete).

Persisters

The following Persisters are found in the Event Driven API:

- CourseSitePersister
- OrganizationPersister
- EnrollmentPersister
- OrganizationMembershipPersister
- StaffAssignmentPersister

© 2005 Blackboard Inc. Proprietary and Confidential

- PersonPersister
- CourseCategoryPersister
- OrganizationCategoryPersister
- CourseCategoryMembershipPersister
- OrganizationCategoryMembershipPersister

Persist methods include changeKey, insert, remove, save, update, clone. Change key is not relevant for membership type items. Clone is only relevant for CourseSite/Organization. More information on Persist methods can be found in the Data Integration API Specification Guide.

Loaders

The following Loaders are found in the Event Driven API:

- CourseSiteLoader
- OrganizationLoader
- EnrollmentLoader
- OrganizationMembershipLoader
- StaffAssignmentLoader
- PersonLoader
- CourseCategoryLoader
- OrganizationCategoryLoader
- CourseCategoryMembershipLoader
- OrganizationCategoryMembershipLoader

Load methods include load by batch_uid and load by template. More information on Load methods can be found in the Data Integration API Specification Guide.

Data Source Loader and Persister

The following Data Source Loader and Persister are found in the Event Driven API:

```
DataSourceLoader
  loadAll()
  loadAdminObjectCount()
  loadAllAdminObjectCounts()
  loadByBatchUid()

DataSourcePersister
  create()
  disableAdminObjectsByDataType()
```

© 2005 Blackboard Inc. Proprietary and Confidential

disableAllAdminObjects()

modify()

purgeAdminObjectsByDataType()

purgeAllAdmiObjects()

purgeSnapshotSessions()

removeByBatchUid()

© 2005 Blackboard Inc. Proprietary and Confidential

DATA INTEGRATION PROCESS

Overview

This topic documents the collaboration between the entity and the persistence classes. This collaboration is the process by which data is obtained from the institution systems; an instance is created for that data, and input into the *Blackboard Learning System* database as a data record.

Relationship between the entity and persister classes

Follow the steps below to determine the relationships between the entity and persistence classes.

1. The program must first initialize the **`BbServiceManager`** object. The method call is **`blackboard.platform.BbServiceManager.init( serviceConfig )`**; where **`serviceConfig`** is a java.io.File object. This object represents a link to the configuration file on the operating system. The file is a detailed list of all services that are active on the instance of the **`servicemanager`** as well as any of their configuration files.

   The **`BbServiceManager`** object is only initialized once for each execution of the application.

   A virtual host is also needed for proper setup. A virtual host can be obtained by first getting the Virtual Installation Manager using **`BBServiceManager.lookupService(VirtualInstallationManager.class).`** The Virtual Installation Manager has a **`getVirtualHost(String id)`** method which returns the virtual host.

   Next, the ContextManager must be retrieved using **`BBServiceManager.lookupService(ContextManager.class)`**. Finally, the context can be set by calling the **`ContextManager's setContext()`** method and passing in the virtual host as an argument.

   The following code sample assumes a "SERVICE_CONFIG_DIR" and "VIRTUAL_HOST_KEY" properties will be set, probably through –D parameters if it is used in a command line application. The SERVICE_CONFIG_DIR should be set to the full path of the Blackboard config directory, while the VIRTUAL_HOST_KEY needs to be the virtual installation you wish to test, by default it is bb_bb60.

```
// Initialize the BbServiceManager and set the context
try
{
blackboard.platform.BbServiceManager.init(System.getProperties().g
etProperty("SERVICE_CONFIG_DIR") + "service-config-snapshot-
jdbc.properties");

// The virtual host is needed to establish the proper database
context.
```

```
VirtualInstallationManager vm = (VirtualInstallationManager)
BbServiceManager.lookupService(VirtualInstallationManager.class);

String vhostUID = System.getProperty("VIRTUAL_HOST_KEY",
"bb_bb60");
VirtualHost vhost = vm.getVirtualHost(vhostUID);

vhost = vm.getVirtualHost(vhostUID);
if(vhost == null)
{
throw new Exception("Virtual Host '" + vhostUID + "' not found.");
}

// Now that the vhost is set we can set the context based on that
vhost
ContextManager cm = (ContextManager)
BbServiceManager.lookupService(ContextManager.class);

Context context = cm.setContext(vhost);
}
catch(Exception e)
{
System.out.println("Exception trying to init the
BbPersistenceManager\n " + e.toString() + "..exiting.\n");
System.exit(0);
}
```

2. The controller creates an entity object and sets its attributes.

   The following is an example of how to get a course site:

```
blackboard.admin.data.course.CourseSite site=new
blackboard.admin.data.course.CourseSite();
```

3. The controller requests a persist action off the Loader / Persister.

```
CourseSiteLoader cLoader=
(CourseSiteLoader)BbServiceManager.getPersistenceService().getDbPe
rsistenceManager().getLoader(CourseSiteLoader.TYPE);
  CourseSitePersister cPersister=
(CourseSitePersister)BbServiceManager.getPersistenceService().getD
bPersistenceManager().getPersister(CourseSitePersister.TYPE);
```

4. The controller catches any persistence exceptions that occur.

© 2005 Blackboard Inc. Proprietary and Confidential

5. Steps 2, 3, and 4 may be repeated for different entities and different persist actions as needed.

© 2005 Blackboard Inc. Proprietary and Confidential

DATA PACKAGE

Overview

The classes in the data package (`blackboard.admin.data`) represent the *Blackboard Learning System* entities from the data model. All entities, including Person, Course/Organization, Enrollment/Staff Assignment/Organization Membership, Course Category/Organization Category, and Course Category Membership/Organization Category Membership, derive from the base class IAdminObject. All data classes are classes containing the fields defined in the data model as well as get/set methods for each.

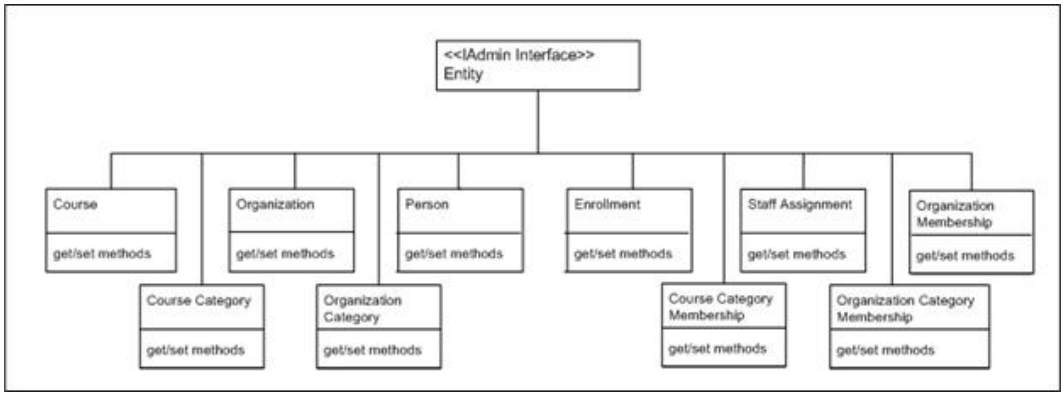
Object model

The object model is based on the information model in the IMS 1.01 Enterprise Specification. The IMS specification provides five basic objects: Person, Course/Organization, Enrollment/Staff Assignment/Organization Membership, Course Category/Organization Category, and Course Category Membership/Organization Category Membership.

Entity classes

The following classes exist within the Entity package. Each class can instantiate objects specific to the *Blackboard Learning System*.

- **Person** represents a user within the system.
- **Course / Organization** are the base abstraction used to build courses or organizations. It is sub-classed to provide additional semantics useful in integration. The provided subclasses are Course and Organization.
- **Enrollment / Staff Assignment / Organization Membership** are objects that represents the association of a Person with a course or organization. **Course Category / Organization Category** data consists of the title, key, and availability information of a catalog category.
- **Course Category Membership / Organization Category Membership** data consists of the information required to connect a course or organization to a category.



Incremental updates

The entity objects support incremental updates. This means that if an entity that already exists in the system needs a single attribute to be updated, then the user of the API needs to only set the entity key and that modified attribute before calling the update() method. This allows a password to be changed without changing everything else.

© 2005 Blackboard Inc. Proprietary and Confidential

PERSISTER PACKAGE

Overview

This topic reviews the persister package found in the Event Driven API. For the most complete and up-to-date information on packages and classes, please refer to the java docs included with the API or the Data Integration Specifications Guide.

Persistence package

The persistence package provides a framework to handle the insert, update, save, and delete for the entity objects. The framework includes:

- **BbServiceManage**r. This class handles the coordination of Blackboard enabled services.
- **PersistenceService**. This class is a singleton, an object only intended to exist once, that manages the different persistence context for the entity objects.
- **DbPersistenceManager**. This class maintains the reference points for each individual loader/persister. An instance of any persister/loader is requested through it.
- **PersisterInterfaces**. This includes the **PersonPersister**, **CourseSitePersister**, etc.

```
blackboard.platform.BbServiceManager.init( serviceConfig,bbprops );

blackboard.platform.persistence.PersistenceService pService =
BbServiceManager.getPersistenceService().

blackboard.persist.BbPersistenceManager bManager =
pService.getDbPersistenceManager()

CourseSiteLoader cLoader=
(CourseSiteLoader)bManager.getLoader(CourseSiteLoader.TYPE);
```

The user must initialize the **BbServiceManager** before attempting any persistence of the admin data objects. After **BbServiceManager** is initialized the appropriate loader/persister must be used according to the data type the programmer intends to manage.

© 2005 Blackboard Inc. Proprietary and Confidential

APPENDIX A – CODE SAMPLES

Overview

This following section includes code samples from the *Blackboard Learning System* Event Driven API.

BLACKBOARD LEARNING SYSTEM EVENT DRIVEN API - DATA OBJECT CREATION

```
blackboard.admin.data.user.Person person_1 = new
blackboard.admin.data.user.Person();

blackboard.admin.data.user.Person person_2 = new
blackboard.admin.data.user.Person();

blackboard.admin.data.course.CourseSite course = new
blackboard.admin.data.course.CourseSite();

blackboard.admin.data.course.Organization org = new
blackboard.admin.data.course.Organization();

blackboard.admin.data.course.Enrollment enroll = new
blackboard.admin.data.course.Enrollment();

blackboard.admin.data.course.OrganizationMembership orgmem = new
blackboard.admin.data.course.OrganizationMembership();

blackboard.admin.data.course.StaffAssignment staff = new
blackboard.admin.data.course.StaffAssignment();

blackboard.admin.data.category.CourseCategory crs_cat = new
blackboard.admin.data.category.CourseCategory();

blackboard.admin.data.category.OrganizationCategory org_cat = new
blackboard.admin.data.category.OrganizationCategory();

blackboard.admin.data.category.CourseCategoryMembership crs_cat_mem = new
blackboard.admin.data.category.CourseCategoryMembership();

blackboard.admin.data.category.OrganizationCategoryMembership org_cat_mem
= new blackboard.admin.data.category.OrganizationCategoryMembership();
```

BLACKBOARD LEARNING SYSTEM EVENT DRIVEN API - DATA OBJECT ATTRIBUTE MARSHALLING

```
person_1.setDataSourceBatchUid("SIS");

person_1.setEmailAddress("hbaal@carthage.com");

person_1.setFamilyName("Baal");

person_1.setGivenName("Hannibal");

person_1.setBatchUid("33333");

person_1.setPassword("hbaal");

person_1.setUserName("hbaal");

person_1.setSystemRole( person_1.SystemRole.SYSTEM_ADMIN);

person_1.setTitle("General");


person_2.setDataSourceBatchUid("SIS");

person_2.setEmailAddress("ahun@carthage.com");

person_2.setFamilyName("Hun");

person_2.setGivenName("Atillah");

person_2.setBatchUid("11111");
```

© 2005 Blackboard Inc. Proprietary and Confidential

```
person_2.setPassword("ahun");
person_2.setUserName("ahun");
person_2.setSystemRole( person_2.SystemRole.SYSTEM_ADMIN);
person_2.setTitle("General");

course.setCourseId("CRTHG101_CRS");
course.setDataSourceBatchUid("SIS");
course.setBatchUid("ACRTHG101_CRS");
course.setTitle("Intimidation through Mammoths");
course.setInstitutionName("Carthage");
course.setDescription("Discussion of the use ");

org.setOrganizationId("CRTHG101_ORG");
org.setDataSourceBatchUid("SIS");
org.setBatchUid("ACRTHG101_ORG");
org.setTitle("Intimidation through Mammoths");
org.setInstitutionName("Carthage");
org.setDescription("Discussion of the use ");

enroll.setDataSourceBatchUid("SIS");
enroll.setCourseSiteBatchUid("ACRTHG101_CRS");
enroll.setPersonBatchUid("33333");

staff.setDataSourceBatchUid("SIS");
staff.setCourseSiteBatchUid("ACRTHG101_CRS");
staff.setPersonBatchUid("11111");
staff.setRole( staff.Role.STUDENT );

orgmem.setDataSourceBatchUid("SIS");
orgmem.setOrganizationBatchUid("ACRTHG101_ORG");
orgmem.setPersonBatchUid("33333");
orgmem.setRole( staff.Role.INSTRUCTOR);

crs_cat.setBatchUid("ACRTHGCAT");
crs_cat.setDataSourceBatchUid("SIS");
crs_cat.setFrontPageInd(true);
crs_cat.setIsAvailable(true);
crs_cat.setTitle("Carthage Course Category");

org_cat.setBatchUid("ACRTHGCAT");
```

© 2005 Blackboard Inc. Proprietary and Confidential

```
org_cat.setDataSourceBatchUid("SIS");
org_cat.setFrontPageInd(true);
org_cat.setIsAvailable(true);
org_cat.setTitle("Carthage Organization Category");

org_cat_mem.setCategoryBatchUid("ACRTHGCAT");
org_cat_mem.setDataSourceBatchUid("SIS");
org_cat_mem.setOrganizationBatchUid("ACRTHG101_ORG");
org_cat_mem.setIsAvailable(true);

crs_cat_mem.setCategoryBatchUid("ACRTHGCAT");
crs_cat_mem.setDataSourceBatchUid("SIS");
crs_cat_mem.setCourseSiteBatchUid("ACRTHG101_CRS");
crs_cat_mem.setIsAvailable(true);
```

BLACKBOARD LEARNING SYSTEM EVENT DRIVEN API - START PERSISTENCE TESTING

```
/*********** Begin testing **********/
try
{
 ePersister.insert(enroll);
}
catch(PersistenceException nfe)
{
 System.out.println("insert enrollment without supporting object(s)
success!");
}

try
{
 saPersister.insert(staff);
}
catch(PersistenceException nfe)
{
 System.out.println("insert staff assignment without supporting object(s)
success!");
}

try
{
```

```
 orgMemPersister.insert(orgmem);
}
catch(PersistenceException nfe)
{
 System.out.println("insert organization membership without supporting
object(s) success!");
}

try
{
 orgCatMemPersister.insert(org_cat_mem);
}
catch(PersistenceException nfe)
{
 System.out.println("insert organization link without supporting
object(s) success!");
}

try
{
 crsCatMemPersister.insert(crs_cat_mem);
}
catch(PersistenceException nfe)
{
 System.out.println("insert course link without supporting object(s)
success!");
}

try
{
 ePersister.save(enroll);
}
catch(PersistenceException nfe)
{
 System.out.println("save enrollment without supporting object(s)
success!");
}

try
{
 saPersister.save(staff);
```

```
}
catch(PersistenceException nfe)
{
 System.out.println("save staff assignment without supporting object(s)
success!");
}


try
{
 orgMemPersister.save(orgmem);
}
catch(PersistenceException nfe)
{
 System.out.println("save organization membership without supporting
object(s) success!");
}


try
{
 orgCatMemPersister.save(org_cat_mem);
}
catch(PersistenceException nfe)
{
 System.out.println("save organization link without supporting object(s)
success!");
}


try
{
 crsCatMemPersister.save(crs_cat_mem);
}
catch(PersistenceException nfe)
{
 System.out.println("save course link without supporting object(s)
success!");
}


try
{
 pPersister.update(person_1);
 pPersister.update(person_2);
```

© 2005 Blackboard Inc. Proprietary and Confidential

```
}
catch(PersistenceException nfe)
{
 System.out.println("Update not found success (Person)!");
}

try
{
 cPersister.update(course);
}
catch(PersistenceException nfe)
{
 System.out.println("Update not found success (Course)!");
}

try
{
 orgPersister.update(org);
}
catch(PersistenceException nfe)
{
 System.out.println("Update not found success (Organization)!");
}

try
{
 ePersister.update(enroll);
}
catch(PersistenceException nfe)
{
 System.out.println("update enrollment without supporting object(s)
success!");
}

try
{
 saPersister.update(staff);
}
catch(PersistenceException nfe)
{
```

```
 System.out.println("update staff assignment without supporting object(s)
success!");
}

try
{
 orgMemPersister.update(orgmem);
}
catch(PersistenceException nfe)
{
 System.out.println("update organization membership without supporting
object(s) success!");
}

try
{
 crsCatPersister.update(crs_cat);
}
catch(PersistenceException nfe)
{
 System.out.println("Update not found success (course Category)!");
}

try
{
 orgCatPersister.update(org_cat);
}
catch(PersistenceException nfe)
{
 System.out.println("Update not found success (organization Category)!");
}

try
{
 orgCatMemPersister.update(org_cat_mem);
}
catch(PersistenceException nfe)
{
 System.out.println("update organization link without supporting
object(s) success!");
}
```

```
try
{
 crsCatMemPersister.update(crs_cat_mem);
}
catch(PersistenceException nfe)
{
 System.out.println("update course link without supporting object(s)
success!");
}


cPersister.insert(course);
System.out.println("course insertion succeeded");

orgPersister.insert(org);
System.out.println("organization insertion succeeded");

pPersister.insert(person_1);
System.out.println("person 1 insertion succeeded");

pPersister.insert(person_2);
System.out.println("person 2 insertion succeeded");

orgMemPersister.insert(orgmem);
System.out.println("org membership insertion succeeded");

saPersister.insert(staff);
System.out.println("staff assignment insertion succeeded");

ePersister.insert(enroll);
System.out.println("enrollment membership insertion succeeded");

crsCatPersister.insert(crs_cat);
System.out.println("course category insertion succeeded");

orgCatPersister.insert(org_cat);
System.out.println("organization category insertion succeeded");

crsCatMemPersister.insert(crs_cat_mem);
```

© 2005 Blackboard Inc. Proprietary and Confidential

```
System.out.println("course link insertion succeeded");

orgCatMemPersister.insert(org_cat_mem);
System.out.println("organization link insertion succeeded");

ePersister.save(enroll);
System.out.println("enroll save succeeded");

saPersister.save(staff);
System.out.println("staff save succeeded");

orgMemPersister.save(orgmem);
System.out.println("org membership save succeeded");

cPersister.save(course);
System.out.println("course save succeeded");

orgPersister.save(org);
System.out.println("organization save succeeded");

pPersister.save(person_1);
System.out.println("person_1 save succeeded");

pPersister.save(person_2);
System.out.println("person_2 save succeeded");

crsCatPersister.save(crs_cat);
System.out.println("course category save succeeded");

orgCatPersister.save(org_cat);
System.out.println("organization category save succeeded");

crsCatMemPersister.save(crs_cat_mem);
System.out.println("course link save succeeded");

orgCatMemPersister.save(org_cat_mem);
System.out.println("organization link save succeeded");

ePersister.remove(enroll);
System.out.println("enroll remove succeeded");
```

© 2005 Blackboard Inc. Proprietary and Confidential

```
saPersister.remove(staff);
System.out.println("staff remove succeeded");

orgMemPersister.remove(orgmem);
System.out.println("org membership remove succeeded");

pPersister.remove(person_1);
System.out.println("person 1 remove succeeded");

pPersister.remove(person_2);
System.out.println("person 2 remove succeeded");

crsCatMemPersister.remove(crs_cat_mem);
System.out.println("course link remove succeeded");

orgCatMemPersister.remove(org_cat_mem);
System.out.println("organization link remove succeeded");

crsCatPersister.remove(crs_cat);
System.out.println("course category remove succeeded");

orgCatPersister.remove(org_cat);
System.out.println("organization category remove succeeded");

cPersister.remove(course);
System.out.println("course remove succeeded");

orgPersister.remove(org);
System.out.println("org remove succeeded");

try
{
 pPersister.remove(person_1);
 pPersister.remove(person_2);
}
catch(PersistenceException nfe)
{
 System.out.println("Remove not found success (Person)!");
}
```

```
try
{
 cPersister.remove(course);
}
catch(PersistenceException nfe)
{
 System.out.println("Remove not found success (Course)!");
}

try
{
 orgPersister.remove(org);
}
catch(PersistenceException nfe)
{
 System.out.println("Remove not found success (Organization)!");
}

try
{
 ePersister.remove(enroll);
}
catch(PersistenceException nfe)
{
 System.out.println("Remove not found success (enroll)!");
}

try
{
 saPersister.remove(staff);
}
catch(PersistenceException nfe)
{
 System.out.println("Remove not found success (staff)!");
}

try
{
 orgMemPersister.remove(orgmem);
```

© 2005 Blackboard Inc. Proprietary and Confidential

```
}
catch(PersistenceException nfe)
{
 System.out.println("Remove not found success (Org Membership)!");
}

try
{
 crsCatPersister.remove(crs_cat);
}
catch(PersistenceException nfe)
{
 System.out.println("Remove not found success (Course Category)!");
}

try
{
 orgCatPersister.remove(org_cat);
}
catch(PersistenceException nfe)
{
 System.out.println("Remove not found success (Organization Category)!");
}

try
{
 crsCatMemPersister.remove(crs_cat_mem);
}
catch(PersistenceException nfe)
{
 System.out.println("Remove not found success (Course link)!");
}

try
{
 orgCatMemPersister.remove(org_cat_mem);
}
catch(PersistenceException nfe)
{
 System.out.println("Remove not found success (Organization link)!");
```

```
}

pPersister.save(person_1);
pPersister.save(person_2);
cPersister.save(course);
orgPersister.save(org);
ePersister.save(enroll);
saPersister.save(staff);
orgMemPersister.save(orgmem);
crsCatPersister.save(crs_cat);
orgCatPersister.save(org_cat);
crsCatMemPersister.save(crs_cat_mem);
orgCatMemPersister.save(org_cat_mem);

System.out.println("Save successful (non-existing)");

/******** Now beginning load test **********************/
java.util.Iterator iter = null;

person_1 = pLoader.load(person_1.getBatchUid());
System.out.println("load of person 1 by batch uid successful");

person_2 = pLoader.load(person_2.getBatchUid());
System.out.println("load of person 2 by batch uid successful");

course = cLoader.load(course.getBatchUid());
System.out.println("load of course by batch uid successful");

org = orgLoader.load(org.getBatchUid());
System.out.println("load of org by batch uid successful");

enroll = eLoader.load(enroll.getCourseSiteBatchUid(),
enroll.getPersonBatchUid() );
System.out.println("load of enroll by batch uid successful");

staff = saLoader.load( staff.getCourseSiteBatchUid(),
staff.getPersonBatchUid() );
System.out.println("load of staff by batch uid successful");

orgmem = orgMemLoader.load( orgmem.getOrganizationBatchUid(),
orgmem.getPersonBatchUid());
```

© 2005 Blackboard Inc. Proprietary and Confidential

```
System.out.println("load of orgmem by batch uid successful");


crs_cat = crsCatLoader.load(crs_cat.getBatchUid());
System.out.println("load of crs cat by batch uid successful");


org_cat = orgCatLoader.load(org_cat.getBatchUid());
System.out.println("load of org cat by batch uid successful");


org_cat_mem = orgCatMemLoader.load(
org_cat_mem.getOrganizationBatchUid(),
org_cat_mem.getCategoryBatchUid());
System.out.println("load of org cat mem by batch uid successful");


crs_cat_mem = crsCatMemLoader.load( crs_cat_mem.getCourseBatchUid(),
crs_cat_mem.getCategoryBatchUid());
System.out.println("load of crs cat mem by batch uid successful");


BbList list = pLoader.load(person_1);


if ( list.size() == 1)
{
 System.out.println("Load of person_1 successful");
}
else throw new Exception("person 1 load failed with " + list.size() + "
count");


list = pLoader.load(person_2);


if ( list.size() == 1)
{
 System.out.println("Load of person 2 successful");
}
else throw new Exception("person 2 load failed with " + list.size() + "
count");


list = cLoader.load(course);


if ( list.size() == 1)
{
 System.out.println("Load of course successful");
}
```

© 2005 Blackboard Inc. Proprietary and Confidential

```
else throw new Exception("course load failed with " + list.size() + "
count");


list = orgLoader.load(org);


if ( list.size() == 1)
{
 System.out.println("Load of org successful");
}
else throw new Exception("org load failed with " + list.size() + "
count");


list = eLoader.load(enroll);


if ( list.size() == 1)
{
 System.out.println("Load of enrollment successful");
}
else
{
 System.out.println(enroll);
 throw new Exception("enrollment load failed with " + list.size() + "
count");
}


list = saLoader.load(staff);
if ( list.size() == 1)
{
 System.out.println("Load of staff successful");
}
else throw new Exception("staff load failed with " + list.size() + "
count");


list = orgMemLoader.load(orgmem);
if ( list.size() == 1)
{
 System.out.println("Load of org membership successful");
}
else throw new Exception("org membership load failed with " + list.size()
+ " count");


list = crsCatMemLoader.load(crs_cat_mem);
```

```
if ( list.size() == 1)
{
 System.out.println("Load of course link successful");
}
else throw new Exception("couse link load failed with " + list.size() + "
count");


list = orgCatMemLoader.load(org_cat_mem);


if ( list.size() == 1)
{
 System.out.println("Load of organization link successful");
}
else throw new Exception("organization link load failed with " +
list.size() + " count");


list = crsCatLoader.load(crs_cat);


if ( list.size() == 1)
{
 System.out.println("Load of course category successful");
}
else throw new Exception("course category load failed with " +
list.size() + " count");


list = orgCatLoader.load(org_cat);


if ( list.size() == 1)
{
 System.out.println("Load of organization category successful");
}
else throw new Exception("organization category load failed with " +
list.size() + " count");


/******************** End load test *************/
crsCatMemPersister.remove(crs_cat_mem);
orgCatMemPersister.remove(org_cat_mem);
crsCatPersister.remove(crs_cat);
orgCatPersister.remove(org_cat);
```

© 2005 Blackboard Inc. Proprietary and Confidential

BLACKBOARD LEARNING SYSTEM EVENT DRIVEN API- GARBAGE CLEAN UP OF DB BOUND OBJECTS

```
try
{
crsCatMemPersister.remove(crs_cat_mem);
}
catch(Exception exc){}
try
{
orgCatMemPersister.remove(org_cat_mem);
}
catch(Exception exc){}
try
{
crsCatPersister.remove(crs_cat);
}
catch(Exception exc){}
try
{
orgCatPersister.remove(org_cat);
}
catch(Exception exc){}

try
{
ePersister.remove(enroll);
}
catch(Exception exc){}
try
{
orgMemPersister.remove(orgmem);
}
catch(Exception exc){}
try
{
saPersister.remove(staff);
}
catch(Exception exc){}
try
```

© 2005 Blackboard Inc. Proprietary and Confidential

```
{
pPersister.remove(person_1);
}
catch(Exception exc){}
try
{
pPersister.remove(person_2);
}
catch(Exception exc){}
try
{
cPersister.remove(course);
}
catch(Exception exc)
{
}
try
{
orgPersister.remove(org);
}
catch(Exception exc)
{
}
```

BLACKBOARD LEARNING SYSTEM EVENT DRIVEN API - START COURSE/ORGANIZATION COPY TESTING (ALL AREAS)

```
/*********** Begin testing **********/

blackboard.admin.data.course.CourseSite tgtCourse = new
blackboard.admin.data.course.CourseSite();

blackboard.admin.data.course.Organization tgtOrg = new
blackboard.admin.data.course.Organization();

blackboard.admin.persist.course.CloneConfig cfg = new
blackboard.admin.persist.course.CloneConfig();


cfg.includeArea( blackboard.admin.persist.course.CloneConfig.Area.ALL );


tgtCourse.setBatchUid(“tgt.course”);

tgtOrg.setBatchUid(“tgt.organization”);


try
```

© 2005 Blackboard Inc. Proprietary and Confidential

```
{
 cPersister.clone(course.getBatchUid(),tgtCourse.getBatchUid(), cfg);
 orgPersister.clone(org.getBatchUid(),tgtOrg.getBatchUid(), cfg);
}
catch(PersistenceException nfe)
{
 System.out.println("insert enrollment without supporting object(s)
success!");
}

ePersister.remove(enroll);
orgMemPersister.remove(orgmem);
saPersister.remove(staff);
pPersister.remove(person_1);
pPersister.remove(person_2);
cPersister.remove(course);
orgPersister.remove(org);

System.out.println("All removals successful (pre-existing)");

System.out.println("Success!");
```

End Persistence Test

APPENDIX B – DATA FORMAT TABLES FOR SNAPSHOT FILES

Overview

This section of the Data Integration Manual provides a series of tables that describe the data feed elements in each data feed file. There are ten types of flat data files that can be input into the *Blackboard Learning System*: user information, course and organization information, enrollment and membership information, staff assignments, course and organization categories, and course and Organization Category Memberships.

The following topics describe the layout of the data in the feed files:

Data tables

Each of the topics that cover data feed formats contains detailed tables that describe each data attribute. The following columns appear in the data feed attribute tables:

Element – the attribute name in Blackboard.

Description and Data Type – the definition of the attribute and the data type of the attribute as it appears in the snapshot feed. Please note that dates can be expressed in yyyymmdd or mm/dd/yyyy format. Beginning with Blackboard Academic Suite (Release 7.0) certain elements accept multi byte characters. These are noted in the tables below.

© 2005 Blackboard Inc. Proprietary and Confidential

USER DATA FEED ELEMENTS

User data

The following fields must be present in a user record for it to be successfully processed:

- EXTERNAL_PERSON_KEY
- USER_ID
- SYSTEM_ROLE
- FIRSTNAME
- LASTNAME
- EMAIL
- INSTITUTION_ROLE

User data table

ELEMENT	DESCRIPTION AND DATA TYPE

ELEMENT	DESCRIPTION AND DATA TYPE
SYSTEM_ROLE	The user's administrative role, describing the user's level of system administration privilege. The role of "none" has no system administration or course creation privileges associated with it, and is the most commonly assigned role. The following strings are acceptable: • User Administrator: "account_admin", "accountadmin" or "user_admin" • System Support: "system_support" or "syssupport" • Course Creator: "course_creator" or "creator" • Support: "course_support" or "support" • Guest: "guest" • None: "none" • Observer: "observer" • Portal Administrator: "portal_admin" or "portal" • System Administrator: "sys_admin", "sysadmin" or "system_admin" • eCommerce Administrator: "ecommerce_admin" • Card Office Administrator: "card_office_admin" • Store Administrator: "store_admin" If one is not given the system defaults to none.
EXTERNAL_PERSON_KEY	A unique identifier for a user at the institution This ID is provided by the institution and is not displayed to users. It is strongly recommended that the key data be a permanent, non-changing identifier for each user. An example of a good key is a permanent student ID assigned when the person applies to the institution. Please note that the database cannot merge data between distinct accounts. String. Max length 64. Multi byte characters accepted. Not Null, External Key
NEW_EXTERNAL_PERSON_KEY	This field is used only on the rare occasions when a user's EXTERNAL_PERSON_KEY changes. String. Max length 64. Multi byte characters accepted.
COMPANY	The name of the company for which the user works. String. Max length 100. Multi byte characters accepted.

© 2005 Blackboard Inc. Proprietary and Confidential

ELEMENT	DESCRIPTION AND DATA TYPE
USER_ID	The username used to log into *Blackboard Learning System*. The user name must be unique. String. Max length 50. Multi byte characters accepted. Not Null
PASSWD	The password used to log into *Blackboard Learning System*. On insert, if a password is not included, than the password defaults to the username (Unless it is "not owned" by the institution). Must be sent MD5 encoded if the encrypt.password property is set to N. The Snapshot controller available with *Blackboard Learning System* is set to handle MD5 encryption. String. Max length 32. Multi byte characters accepted. If multi byte characters are included, the encrypt.password property must be set to Y.
STUDENT_ID	Generic identifier field. This is a display only field. String. Max length 100. Multi byte characters accepted.
EMAIL	The email address of the user. Blackboard recommends that user_email is not null. Sending email without the From field populated can cause problems. String. Max length 100
STREET_1	The first line of the address of the user. String. Max length 100. Multi byte characters accepted.
STREET_2	The second line of the address of the user. String. Max length 100. Multi byte characters accepted.
GENDER	The gender of the user. Passed as a string. ("Not Disclosed", "Male", "Female")
BIRTHDATE	The user's birth date in yyyymmdd format. Date
TITLE	The title that the user prefers to use. String. Max length 100. Multi byte characters accepted.
CITY	The city name corresponding to the user's address. String. Max length 50. Multi byte characters accepted.

© 2005 Blackboard Inc. Proprietary and Confidential

ELEMENT	DESCRIPTION AND DATA TYPE
STATE	The state or province name corresponding to the user's address. String. Max length 50. Multi byte characters accepted.
ZIP_CODE	The user's ZIP or Postal Code. String. Max length 50. Multi byte characters accepted.
DEPARTMENT	The name of the department or sub-section where the user works. String. Max length 100. Multi byte characters accepted.
COUNTRY	The country name corresponding to the user's address. String. Max length 50. Multi byte characters accepted.
B_PHONE_1	The work phone number associated with the user. String. Max length 50. Multi byte characters accepted.
B_PHONE_2	An additional work phone number associated with the user. String. Max length 50. Multi byte characters accepted.
FIRSTNAME	The user's given first name. String. Max length 100. Multi byte characters accepted. Not null
H_FAX	The user's home fax number. String. Max length 50. Multi byte characters accepted.
B_FAX	The work fax number associated with the user. String. Max length 50. Multi byte characters accepted.
H_PHONE_1	The user's home phone number. String. Max length 50. Multi byte characters accepted.
H_PHONE_2	An additional home phone number associated with the user. String. Max length 50. Multi byte characters accepted.
M_PHONE	The user's mobile phone or pager number. String. Max length 50. Multi byte characters accepted.

© 2005 Blackboard Inc. Proprietary and Confidential

ELEMENT	DESCRIPTION AND DATA TYPE
JOB_TITLE	The associated job title for the user. String. Max length 100. Multi byte characters accepted.
PUBLIC_IND	Determines if the user's personal information will be displayed in the User Directory. Values may be "Y" for yes, or "N" for no. Char(1). Y/N
AVAILABLE_IND	User account availability within the user interface. If set to no, the user account will appear in gray to the Administrator and Instructors. The user will not be able to log into the system. Values may be "Y" for yes, or "N" for no. Char(1). Y/N
ADDRESS_IND	Determines if the user's home address will be displayed in the User Directory. Values may be "Y" for yes, or "N" for no. If a value is not provided the default is no. Char(1). Y/N
EMAIL_IND	Determines if the user's email address will be displayed in the User Directory. Values may be "Y" for yes, or "N" for no. If a value is not provided the default is no. Char(1). Y/N
PHONE_IND	Determines if home phone, fax or cell phone numbers will be displayed in the User Directory. Values may be "Y" for yes, or "N" for no. If a value is not provided the default is no. Char(1). Y/N
WORK_IND	Determines if work contact information (company, department, title, phone, fax) will be displayed in the User Directory. Values may be "Y" for yes, or "N" for no. If a value is not provided the default is no. Char(1). Y/N
LASTNAME	The user's family/last name. String. Max length 100. Multi byte characters accepted. Not null
MIDDLENAME	The user's given middle name. String. Max length 100. Multi byte characters accepted.

© 2005 Blackboard Inc. Proprietary and Confidential

ELEMENT	DESCRIPTION AND DATA TYPE
INSTITUTION_ROLE	The user's Primary Institution Role, which determines the user's view of the Portal Modules and Academic Web Resources. One of the following strings: "Student", "Faculty", "Staff", "Alumni", "Observer", "ProspectiveStudent", "Guest", "Other".
ROW_STATUS	Sets the value of the record to one of the following: Enabled: Normal access to the record. Disabled: Record is visible in some areas of the UI, but may not be changed or accessed. Deleted: Record is scheduled to be removed. Passed as a string. ("enabled", "disabled", "deleted")
EDUC_LEVEL	The highest level of education achieved by the user. Passed as a string. ("K-8", "high school", "freshman", "sophomore", "junior", "senior", "graduate school", "post-graduate school")
WEBPAGE	The URL of the user's personal Web page, if they have one. String. Max length 100
NEW_DATA_SOURCE_KEY	Key used to establish grouping of user elements. Passed as a string. Multi byte characters accepted.
CARD_NUMBER	Identifier for a Campus Card used with the Blackboard Transaction System. Passed as a string or as an encrypted string depending on the encrypt.card.number property.
LOCALE	Identifier for the user's preferred language pack. The value is expressed as XX_xx, for example, the French language pack is represented by FR_fr.

© 2005 Blackboard Inc. Proprietary and Confidential

COURSE AND ORGANIZATION DATA ELEMENTS

Course data

Course information files and organization information files are almost identical and, for this reason, are grouped together and described in the Course and Organization Data Feed Elements table. The data elements are, in most cases, exactly the same for courses and organizations.

The following fields are required to successfully process a course record:

- COURSE_ID
- EXTERNAL_COURSE_KEY
- COURSE_NAME

Warning: Do not mix course and organization data together in the same feed file.

Course and Organization data table

The table below details the course data feed elements. Please note that, where applicable, the organization-specific data element is included under the course data element.

ELEMENT	DESCRIPTION AND DATA TYPE
COURSE_ID ORGANIZATION_ID	Short name used by the institution to uniquely identify the course (for example, math101 _F99) The following characters may not be used in a course ID: " ()&/'+" Not null String. Max length 50. Must be unique and cannot be changed.
EXTERNAL_COURSE_KEY EXTERNAL_ORGANIZATION_KEY	Short name used by the Institution to uniquely identify the course section. This is not displayed to users. Can contain only letters, digits, dashes and periods. No spaces or other punctuation allowed. May be concatenation of Course_ID, Section_ID, and Term. Must be unique for each course, and is often the same as the COURSE_ID. Not null, External key. String. Max length 64. Must be unique. Multi byte characters accepted.

© 2005 Blackboard Inc. Proprietary and Confidential

ELEMENT	DESCRIPTION AND DATA TYPE
NEW_EXTERNAL_COURSE_KEY NEW_EXTERNAL_ORGANIZATION_KEY	This field is used only on the rare occasions when a course's EXTERNAL_COURSE_KEY changes. String. Max length 64. Must be unique. Multi byte characters accepted.
COURSE_NAME ORGANIZATION_NAME	Complete title of the course. The course name is used when sorting. String. Max length 255. Multi byte characters accepted. Not null
ALLOW_GUEST_IND	Allows guest access to the course. String. y/n
DESCRIPTION	Complete description of the course. String. Max length 4000. Multi byte characters accepted.
END_DATE	Date on which access to the course ends. Date (yyyymmdd).
START_DATE	Date on which access to the course section begins. Date (yyyymmdd).
NEW_DATA_SOURCE_KEY	Key used to establish a grouping of course or organization elements. String. Multi byte characters accepted.
ROW_STATUS	Sets the value of the record to one of the following: Enabled: Normal access to the record. Disabled: Record is visible in some areas of the UI, but may not be changed or accessed. Deleted: Record is scheduled to be removed. Passed as a string. ("enabled", "disabled", "deleted")
AVAILABLE_IND	Establishes course or organization availability to Blackboard. Char(1). Y/N
CATALOG_IND	Establishes whether the course or organization appears in catalog. Char(1). Y/N
DESC_PAGE	Determines whether or not to display description information in the course or organization catalog. Char(1). Y/N

© 2005 Blackboard Inc. Proprietary and Confidential

ELEMENT	DESCRIPTION AND DATA TYPE
LOCKOUT_IND	Indicates if access to the course or organization has been restricted. If set to “Y” access to the course or organization will be restricted based on the END_DATE and START_DATE. Char(1). Y/N
PACE	Indicates if the course is instructor-led or self-paced according to the student. Char(1). Mapping accepted {“Self”, “Instructor”}
ABS_LIMIT	Handles absolute limit on content in bytes. Numeric.
SOFT_LIMIT	Handles soft limit on content in bytes. Numeric.
UPLOAD_LIMIT	Handles limit on uploads in bytes. Numeric.
ENROLL_START	Date that enrollment may begin. Date (yyyymmdd).
ENROLL_END	Date that enrollment is no longer available to Students. Date (yyyymmdd).
ENROLL_OPTION	Determines the enrollment method. One of the following strings: “instructor”, “self”, or “email”.
DAYS_OF_USE	Number of days that Students may access the course after enrollment. Useful for self-paced learning. Numeric.
DURATION	Schedules enrollment window. Options are: Continuous: The Course is always accessible. Range: The Course is accessible in the days between one date and another. Either the beginning date or the end date can be left open-ended to make a course accessible from a certain date or until a certain date. Fixed: The Course is accessible for a set number of days. Char(1). Mapping accepted {“Continuous”, “Range”, “Fixed” }
INSTITUTION_NAME	The name of the institution. String. Max length 255. Multi byte characters accepted.

© 2005 Blackboard Inc. Proprietary and Confidential

ELEMENT	DESCRIPTION AND DATA TYPE
CLASSIFICATION_KEY	Sting-based key that establishes an association with a subject area and discipline within the Blackboard Resource Center.
TEMPLATE_COURSE_KEY TEMPLATE_ORGANIZATION_KEY	Designates content source for copy operation. String. Max length 64. Multi byte characters accepted.
LOCALE	Identifier for the course language pack. The value is expressed as XX_xx, for example, the French language pack is represented by FR_fr.
LOCALE_ENFORCED	A Y/N flag that determines if the language pack is enforced when a user accesses the course. If the langauage pack is not enforced, the user may view the course using their preferred language pack.

© 2005 Blackboard Inc. Proprietary and Confidential

ENROLLMENT AND STAFF ASSIGNMENTS DATA FEED ELEMENTS

Membership data

Membership data includes two subclasses: Enrollment and Staff Assignments. Enrollment requires a **ROLE** of Student, while Staff Assignments require any role including Student. Each subclass, enrollment and staff assignments, should be input in separate feed files. Including a user with a role other than Student or Guest in an enrollment (**ENR_SNPSHT**) will cause that record to fail.

Enrollment information files can also contain organization membership information. The primary key for enrollments is a composite of the user ID and the course or organization ID. Enrollment files may only add users to a course or organization as a student or member respectively. Staff assignments must be input in a separate file that can contain staff assignments for courses and organizations.

Enrollment and Staff Assignments data table

The table below details the Enrollment and Staff Assignments data feed elements.

ELEMENT	DESCRIPTION AND DATA TYPE
EXTERNAL_COURSE_KEY EXTERNAL_ORGANIZATION_KEY	Used to look up appropriate keys. This is the same key used to identify the course or organization. This is the same key as the course. Not Null, external key. Multi byte characters accepted.
EXTERNAL_PERSON_KEY	Used to look up appropriate keys. This is the same key used to identify the user. Not Null, external key. Multi byte characters accepted.
ROLE	The user's role in the course. String. This value is mapped to a one character database constant. The valid values for the course role are: instructor, teaching_assistant, course_builder, grader, student, guest, none. Not Null.
ROW_STATUS	Sets the value of the record to one of the following: Enabled: Normal access to the record. Disabled: Record is visible in some areas of the UI, but may not be changed or accessed. Deleted: Record is scheduled to be removed. Passed as a string. ("enabled", "disabled", "deleted")

© 2005 Blackboard Inc. Proprietary and Confidential

ELEMENT	DESCRIPTION AND DATA TYPE
AVAILABLE_IND	Establishes enrollment availability in *Blackboard Learning System*. Char(1). Y/N
LINK_NAME_1	Link name for a user's home page within a course. Each user may have a maximum of three home pages within a course. String. Max length 100. Multi byte characters accepted.
LINK_URL_1	Link URL for a user's home page within a course. Each user may have a maximum of three home pages within a course. String. Max length 100
LINK_DESC_1	Link description for a user's home page within a course. Each user may have a maximum of three home pages within a course. String. Max length 255. Multi byte characters accepted.
LINK_NAME_2	Link name for a user's home page within a course. Each user may have a maximum of three home pages within a course. String. Max length 100. Multi byte characters accepted.
LINK_URL_2	Link URL for a user's home page within a course. Each user may have a maximum of three home pages within a course. String. Max length 100
LINK_DESC_2	Link description for a user's home page within a course. Each user may have a maximum of three home pages within a course. String. Max length 255. Multi byte characters accepted.
LINK_NAME_3	Link name for a user's home page within a course. Each user may have a maximum of three home pages within a course. String. Max length 100. Multi byte characters accepted.
LINK_URL_3	Link URL for a user's home page within a course. Each user may have a maximum of three home pages within a course. String. Max length 100
LINK_DESC_3	Link description for a user's home page within a course. Each user may have a maximum of three home pages within a course. String. Max length 255. Multi byte characters accepted.

© 2005 Blackboard Inc. Proprietary and Confidential

ELEMENT	DESCRIPTION AND DATA TYPE
INTRO	Introduction information for User Home Page within the course. String. Max length 4000. Multi byte characters accepted.
PINFO	Private Information for User Home Page. String. Unlimited. Multi byte characters accepted.
NEW_DATA_SOURCE_KEY	Key used to establish grouping of enrollment elements. Multi byte characters accepted.

© 2005 Blackboard Inc. Proprietary and Confidential

CATEGORY DATA FEED ELEMENTS

Category data

Category data includes information on the groups that create a catalog structure to course or organization listings. Within the *Blackboard Learning System*, courses and organizations are listed under a category within the course catalog or the organization catalog. The date elements below allow an institution to map category information from an external course to the *Blackboard Learning System*. The link data elements, presented in the next table, then assign courses or organizations to a category.

Category data table

The table below details the category data elements.

ELEMENT	DESCRIPTION AND DATA TYPE
EXTERNAL_CATEGORY_KEY	Must be unique. Please note that this field corresponds to the Category Mnemonic field in the UI. Not null. String. Max length 64. Multi byte characters accepted.
ROW_STATUS	Sets the value of the record to one of the following: Enabled: Normal access to the record. Disabled: Record is visible in some areas of the UI, but may not be changed or accessed. Deleted: Record is scheduled to be removed. Passed as a string. (“enabled”, “disabled”, “deleted”)
AVAILABLE_IND	Establishes category availability within *Blackboard Learning System*. Char(1). Y/N
TITLE	The name of the category as it will appear to users within the UI. String. Max length 255. Multi byte characters accepted.
FRONTPAGE_IND	Determines whether or not the category is displayed on the front page of the catalog. Char(1). Y/N
NEW_DATA_SOURCE_KEY	Key used to establish grouping of category elements. String based key. Multi byte characters accepted.

© 2005 Blackboard Inc. Proprietary and Confidential

ELEMENT	DESCRIPTION AND DATA TYPE
PARENT_CATEGORY_KEY	Identifies the parent category in the category tree. String based key. Multi byte characters accepted.
NEW_EXTERNAL_CATEGORY_KEY	Designates replacement for the current key. String. Max length 64. Multi byte characters accepted.

© 2005 Blackboard Inc. Proprietary and Confidential

CATEGORY MEMBERSHIP DATA FEED ELEMENTS

Link data

Category Membership data creates an association between a course or organization and a category.

Category Membership data table

The table below details the link data elements.

ELEMENT	DESCRIPTION AND DATA TYPE
EXTERNAL_CATEGORY_KEY	String. Max length 64. Multi byte characters accepted. Not null
EXTERNAL_COURSE_KEY / EXTERNAL_ORGANIZATION_KEY	Internal numeric Id. Multi byte characters accepted. Not null
ROW_STATUS	Sets the value of the record to one of the following: Enabled: Normal access to the record. Disabled: Record is visible in some areas of the UI, but may not be changed or accessed. Deleted: Record is scheduled to be removed. Passed as a string. (“enabled”, “disabled”, “deleted”)
AVAILABLE_IND	Establishes the Category Membership availability within *Blackboard Learning System*. Char(1). Y/N
NEW_DATA_SOURCE_KEY	Designates replacement for the current key. String. Max length 64. Multi byte characters accepted.

© 2005 Blackboard Inc. Proprietary and Confidential

USER INSTITUTION ROLE MEMBERSHIP DATA FEED ELEMENTS

User Institution Role Membership data

To match a user to multiple Institution Roles requires multiple records in the file. The table below outlines the fields included in a record in a Snapshot User Institution Role Membership file.

User Institution Role Membership data table

ELEMENT	DESCRIPTION
EXTERNAL_PERSON_KEY	A unique identifier for a user at the institution This ID is provided by the institution and is not displayed to users. String. Max length 64. Multi byte characters accepted. Not Null, External Key
ROLE_ID	Institution Role. Passed as a string. Not null. Multi byte characters accepted.

© 2005 Blackboard Inc. Proprietary and Confidential

APPENDIX C DATA FORMAT TABLES FOR XML FILES

Overview

This section of the Data Integration Manual provides a series of tables that describe the data feed elements and attributes in each XML file.

Data tables

Each of the topics that cover data feed formats contains detailed tables that describe each data attribute. The following columns appear in the data feed attribute tables:

- **XML Construction** the XML hierarchy that defines each piece of data.
- **Description and Data Type** the definition of the data and the formatting. Beginning with Blackboard Academic Suite (Release 7.0) certain elements accept multi byte characters. These are noted in the tables below.

IMS Standard

XML files processed by Snapshot conform to the IMS standard. Additional data, specific only to Blackboard, is handled by extending the IMS standard with the element **<EXTENSION>.** Blackboard supports both the IMS 1.0.92 and the more recent IMS 1.1 standard.

The tables below show how to construct XML elements and attributes to handle Blackboard data. Please note that these examples reflect the IMS 1.0.92 standard. In some examples, the equivalent IMS 1.1 standard construction is also given.

Please see http://www.imsglobal.org for more information on the IMS Standard.

Create an XML File for Snapshot

Begin the XML file with the <ENTERPRISE> element. It is possible to include different data types, such as course data, user data, and category membership data in the same XML file.

Begin by encapsulating each item with one of the following sets of tags:

- User: <PERSON></PERSON>
- Course or Organization: <GROUP></GROUP>
- Course Membership or Organization Membership: <MEMBERSHIP></MEMBERSHIP>
- Course Category or Organization Category: <CATEGORY>
- Course Category Membership or Organization Category Membership: <CATEGORY_MEMBERSHIP></CATEGORY_MEMBERSHIP>

Use the data definitions in the following tables to complete each item.

XML User Data <PERSON>

© 2005 Blackboard Inc. Proprietary and Confidential

XML CONSTRUCTION	DESCRIPTION
<EXTENSION> <X_BB_SYSTEMROLE>	The users administrative role, describing the users level of system administration privilege. The role of none has no system administration or course creation privileges associated with it, and is the most commonly assigned role. The following numeric values are acceptable: 0 – System Administrator 1 – System Support 2 – Course Creator 3 – Account Administrator 4 – None 5 – Course Support 6 – User 7 – Observer 8 – Guest 9 – Integration (No GUI access) 10 – Portal Administrator 11 – eCommerce Administrator 12 – Card Office Administrator 13 – eMarketplace Administrator If one is not given the system defaults to none.
<SOURCEDID> <ID>	A unique identifier for a user at the institution This ID is provided by the institution and is not displayed to users. It is strongly recommended that the key data be a permanent, non-changing identifier for each user. An example of a good key is a permanent student ID assigned when the person applies to the institution. Please note that the database cannot merge data between distinct accounts. String. Max length 64. Multi byte characters accepted. Not Null, External Key
<EXTENSION> <X_BB_REPLACEMENTKEY>	This field is used only on the rare occasions when a users EXTERNAL_PERSON_KEY changes. String. Max length 64. Multi byte characters accepted.
n/a	The name of the company for which the user works. String. Max length 100. Multi byte characters accepted.

© 2005 Blackboard Inc. Proprietary and Confidential

XML CONSTRUCTION	DESCRIPTION
<SOURCEDID> <USERID>	The username used to log into *Blackboard Learning System*. The user name must be unique. String. Max length 50. Multi byte characters accepted. Not Null
<EXTENSION> <X_BB_PASSWORD>	The password used to log into *Blackboard Learning System*. On insert, if a password is not included, than the password defaults to the username (Unless it is not owned by the institution). Must be sent MD5 encoded if the encrypt.password property is set to N. The Snapshot controller available with *Blackboard Learning System* is set to handle MD5 encryption. String. Max length 32. Multi byte characters accepted. If multi byte characters are used, the encrypt.password property must be set to N. The IMS 1.1 standard uses <USERID password= > to express a password.
<EXTENSION> <X_BB_STUDENTID>	Generic identifier field. This is a display only field. String. Max length 100. Multi byte characters accepted.
<EMAIL>	The email address of the user. Blackboard recommends that user_email is not null. Sending email without the **From** field populated can cause problems. String. Max length 100
<ADR> <STREET>	The first line of the address of the user. String. Max length 100. Multi byte characters accepted.
<ADR> <STREET>	The second line of the address of the user. String. Max length 100. Multi byte characters accepted.
<DEMOGRAPHICS> <GENDER>	The gender of the user. 0=unknown 1=female 2=male
<DEMOGRAPHICS> <BDAY>	The users birth date in yyyymmdd format. Date

© 2005 Blackboard Inc. Proprietary and Confidential

XML CONSTRUCTION	DESCRIPTION
<NAME> <N> <PREFIX>	The title that the user prefers to use. String. Max length 100. Multi byte characters accepted.
<ADR> <LOCALITY>	The city name corresponding to the users address. String. Max length 50. Multi byte characters accepted.
<ADR> <REGION>	The state or province name corresponding to the users address. String. Max length 50. Multi byte characters accepted.
<ADR> <PCODE>	The users ZIP or Postal Code. String. Max length 50. Multi byte characters accepted.
n/a	The name of the department or sub-section where the user works. String. Max length 100. Multi byte characters accepted.
<ADR> <COUNTRY>	The country name corresponding to the users address. String. Max length 50. Multi byte characters accepted.
<TEL> <TEL teletype="2">	The work phone number associated with the user. String. Max length 50. Multi byte characters accepted.
<TEL> <TEL teletype="6">	An additional work phone number associated with the user. String. Max length 50. Multi byte characters accepted.
<NAME> <N> <GIVEN>	The users given first name. String. Max length 100 Not null. Multi byte characters accepted.
<TEL> <TEL teletype="1">	The users home fax number. String. Max length 50. Multi byte characters accepted.
<TEL> <TEL teletype="3">	The work fax number associated with the user. String. Max length 50. Multi byte characters accepted.
<TEL> <TEL teletype="0">	The users home phone number. String. Max length 50. Multi byte characters accepted.

© 2005 Blackboard Inc. Proprietary and Confidential

XML CONSTRUCTION	DESCRIPTION
<TEL> <TEL teletype="5">	An additional home phone number associated with the user. String. Max length 50. Multi byte characters accepted.
<TEL> <TEL teletype="4">	The users mobile phone or pager number. String. Max length 50. Multi byte characters accepted.
n/a	The associated job title for the user. String. Max length 100. Multi byte characters accepted.
<EXTENSION> <X_BB_PUBLIC_INDICATOR>	Determines if the users personal information will be displayed in the User Directory. Values may be Y for yes, or N for no. Char(1). Y/N
<EXTENSION> <X_BB_AVAILABLE>	User account availability within the user interface. If set to no, the user account will appear in gray to the Administrator and Instructors. The user will not be able to log into the system. Values may be Y for yes, or N for no. Char(1). Y/N
<EXTENSION> <X_BB_ADDRESS_INDICATOR>	Determines if the users home address will be displayed in the User Directory. Values may be Y for yes, or N for no. If a value is not provided the default is no. Char(1). Y/N
<EXTENSION> <X_BB_EMAIL_INDICATOR>	Determines if the users email address will be displayed in the User Directory. Values may be Y for yes, or N for no. If a value is not provided the default is no. Char(1). Y/N
<EXTENSION> <X_BB_CONTACT_INDICATOR>	Determines if home phone, fax or cell phone numbers will be displayed in the User Directory. Values may be Y for yes, or N for no. If a value is not provided the default is no. Char(1). Y/N
<EXTENSION> <X_BB_WORK_INDICATOR>	Determines if work contact information (company, department, title, phone, fax) will be displayed in the User Directory. Values may be Y for yes, or N for no. If a value is not provided the default is no. Char(1). Y/N

© 2005 Blackboard Inc. Proprietary and Confidential

XML Construction	Description
<NAME> <N> <FAMILY>	The users family/last name. String. Max length 100 Not null. Multi byte characters accepted.
<NAME> <N> <OTHER>	The user's given middle name. String. Max length 100. Multi byte characters accepted.
<EXTENSION> <X_BB_INSTITUTIONROLE>	The users Primary Institution Role, which determines the users view of the Portal Modules and Academic Web Resources. One of the following strings: Student, Faculty, Staff, Alumni, Observer, ProspectiveStudent, Guest, Other. The IMS 1.1 standard supports the <INSTITUTIONROLE> element.
<EXTENSION> <X_BB_ROW_STATUS>	Sets the value of the record to one of the following: Enabled: Normal access to the record. Disabled: Record is visible in some areas of the UI, but may not be changed or accessed. Deleted: Record is scheduled to be removed. Passed as a string. (enabled, disabled, deleted)
<EXTENSION> <X_BB_DATASOURCE_KEY>	Key used to establish grouping of user elements. Passed as a string. Multi byte characters accepted.
<EXTENSION> <X_BB_CARD_NUMBER>	Identifier for a Campus Card used with the Blackboard Transaction System. Passed as a string or as an encrypted string depending on the encrypt.card.number property.
<EXTENSION> <X_BB_LOCALE>	Identifier for the preferred user language pack. The value is expressed as XX_xx, for example, the French language pack is represented by FR_fr.

Course and Organization Data <GROUP>

XML Element	Description

XML Element	Description
<DESCRIPTION> <SHORT>	Short name used by the institution to uniquely identify the course (for example, math101 _F99) The following characters may not be used in a course ID: " ()&/'+" Not null String. Max length 50. Must be unique and cannot be changed.
<SOURCEDID> <ID>	Short name used by the Institution to uniquely identify the course section. This is not displayed to users. Can contain only letters, digits, dashes and periods. No spaces or other punctuation allowed. May be concatenation of Course_ID, Section_ID, and Term. Must be unique for each course, and is often the same as the COURSE_ID. Not null, External key. String. Max length 64. Must be unique. Multi byte characters accepted.
<EXTENSION> <X_BB_REPLACEMENTKEY>	This field is used only on the rare occasions when a courses EXTERNAL_COURSE_KEY changes String. Max length 64. Must be unique. Multi byte characters accepted.
<DESCRIPTION> <LONG>	Complete title of the course. The course name is used when sorting. String. Max length 255. Multi byte characters accepted. Not null
<X_BB_GUEST_INDICATOR>	Allows guest access to the course. String. y/n
<DESCRIPTION> <FULL>	Complete description of the course. String. Max length 4000. Multi byte characters accepted.
<TIMEFRAME> <END>	Date on which access to the course ends. Date (yyyymmdd).
<TIMEFRAME> <BEGIN>	Date on which access to the course section begins. Date (yyyymmdd).

© 2005 Blackboard Inc. Proprietary and Confidential

XML Element	Description
<EXTENSION> <X_BB_DATASOURCE_KEY>	Key used to establish a grouping of course or organization elements. String. Multi byte characters accepted.
<EXTENSION> <X_BB_ROW_STATUS>	Sets the value of the record to one of the following: Enabled: Normal access to the record. Disabled: Record is visible in some areas of the UI, but may not be changed or accessed. Deleted: Record is scheduled to be removed. Passed as a string. (enabled, disabled, deleted)
<EXTENSION> <X_BB_AVAILABLE>	Establishes course or organization availability to Blackboard. Char(1). Y/N
<EXTENSION> <X_BB_CATALOG>	Establishes whether the course or organization appears in catalog. Char(1). Y/N
<EXTENSION> <X_BB_DESCRIPTION_PAGE>	Determines whether or not to display description information in the course or organization catalog. Char(1). Y/N
<EXTENSION> <X_BB_LOCKOUT_INDICATOR>	Indicates if access to the course or organization has been restricted. If set to Y access to the course or organization will be restricted based on the END_DATE and START_DATE. Char(1). Y/N
<EXTENSION> <X_BB_PACE>	Indicates if the course is instructor-led or self-paced according to the student. Char(1). Mapping accepted {Self, Instructor}
<EXTENSION> <X_BB_ABSOLUTE_LIMIT>	Handles absolute limit on content in bytes. Numeric.
<EXTENSION> <X_BB_SOFT_LIMIT>	Handles soft limit on content in bytes. Numeric.
<EXTENSION> <X_BB_UPLOAD_LIMIT>	Handles limit on uploads in bytes. Numeric.

© 2005 Blackboard Inc. Proprietary and Confidential

XML Element	Description
<EXTENSION> <X_BB_ENROLL_START>	Date that enrollment may begin. Date (yyyymmdd).
<EXTENSION> <X_BB_ENROLL_END>	Date that enrollment is no longer available to Students. Date (yyyymmdd).
<EXTENSION> <X_BB_ENROLLMENT_TYPE>	Determines the enrollment method. One of the following strings: instructor, self, or email.
<EXTENSION> <X_BB_DAYS_OF_USE>	Number of days that Students may access the course after enrollment. Useful for self-paced learning. Numeric.
<EXTENSION> <X_BB_DURATION>	Schedules enrollment window. Options are: Continuous: The Course is always accessible. Range: The Course is accessible in the days between one date and another. Either the beginning date or the end date can be left open-ended to make a course accessible from a certain date or until a certain date. Fixed: The Course is accessible for a set number of days. Char(1). Mapping accepted {Continuous, Range, Fixed }
<EXTENSION> <X_BB_INSTITUTION_NAME>	The name of the institution. String. Max length 255. Multi byte characters accepted.
<EXTENSION> <X_BB_CLASSIFICATIONKEY>	Sting-based key that establishes an association with a subject area and discipline within the Blackboard Resource Center.
<EXTENSION> <X_BB_TEMPLATEKEY>	Designates content source for copy operation. String. Max length 64. . Multi byte characters accepted.
<EXTENSION> <X_BB_GROUP_TYPE>	0-Course 1-Organization
<EXTENSION> <X_BB_LOCALE>	Identifier for the course language pack. The value is expressed as XX_xx, for example, the French language pack is represented by FR_fr.

© 2005 Blackboard Inc. Proprietary and Confidential

XML Element	Description
<EXTENSION> <X_BB_LOCALE_ENFORCED_INDICATOR>	A Y/N flag that determines if the language pack is enforced when a user accesses the course. If the langauage pack is not enforced, the user may view the course using their preferred language pack.

Enrollment and Staff Assignments Data <MEMBERSHIP>

XML Element	Description and Data Type
<SOURCEDID> <ID>	Used to look up appropriate keys. This is the same key used to identify the course or organization. This is the same key as the course. Not Null, external key. Multi byte characters accepted.
<MEMBER> <SOURCEDID> <ID>	Used to look up appropriate keys. This is the same key used to identify the user. Not Null, external key. Multi byte characters accepted.
<MEMBER> <ROLE>	The users role in the course. String. This value is mapped to a one character database constant. The valid values for the course role are: instructor, teaching_assistant, course_builder, grader, student, guest, none. Not Null
<MEMBER> <ROLE> <STATUS>	Sets the value of the record to one of the following: Enabled: Normal access to the record. Disabled: Record is visible in some areas of the UI, but may not be changed or accessed. Deleted: Record is scheduled to be removed. Passed as a string. (enabled, disabled, deleted)
<EXTENSION> <BB_REPLACEMENTKEY>	Establishes enrollment availability in *Blackboard Learning System*. Char(1). Y/N
<EXTENSION> <X_BB_DATASOURCE_KEY>	Key used to establish grouping of enrollment elements. Multi byte characters accepted.

© 2005 Blackboard Inc. Proprietary and Confidential

Category Data <CATEGORY>

XML Construction	Description
<SOURCEDID> <ID>	Must be unique. Please note that this field corresponds to the Category Mnemonic field in the UI. Not null. String. Max length 64. Multi byte characters accepted.
<EXTENSION> <X_BB_ROW_STATUS>	Sets the value of the record to one of the following: Enabled: Normal access to the record. Disabled: Record is visible in some areas of the UI, but may not be changed or accessed. Deleted: Record is scheduled to be removed. Passed as a string. (enabled, disabled, deleted)
<EXTENSION> <X_BB_AVAILABLE>	Establishes category availability within *Blackboard Learning System*. Char(1). Y/N
<EXTENSION> <X_BB_TITLE>	The name of the category as it will appear to users within the UI. String. Max length 255. Multi byte characters accepted.
<EXTENSION> <X_BB_FRONTPAGE_INDICATOR>	Determines whether or not the category is displayed on the front page of the catalog. Char(1). Y/N
<EXTENSION> <X_BB_DATASOURCE_KEY>	Key used to establish grouping of category elements. String based key. Multi byte characters accepted.
<EXTENSION> <X_BB_PARENT_KEY>	Identifies the parent category in the category tree. String based key. Multi byte characters accepted.
<EXTENSION> <X_BB_REPLACEMENT_KEY>	Designates replacement for the current key. String. Max length 64. Multi byte characters accepted.

Category Membership Data <CATEGORY_MEMBERSHIP>

XML Construction	Description

© 2005 Blackboard Inc. Proprietary and Confidential

XML Construction	Description
<MEMBER> <SOURCEDID> <ID>	String. Max length 64. Not null. Multi byte characters accepted.
<SOURCEDID> <ID>	Internal numeric Id. Not null. Multi byte characters accepted.
<EXTENSION> <X_BB_ROW_STATUS>	Sets the value of the record to one of the following: Enabled: Normal access to the record. Disabled: Record is visible in some areas of the UI, but may not be changed or accessed. Deleted: Record is scheduled to be removed. Passed as a string. (enabled, disabled, deleted)
<EXTENSION> <X_BB_AVAILABLE>	Establishes the Category Membership availability within *Blackboard Learning System*. Char(1). Y/N
<EXTENSION> <X_BB_DATASOURCE_KEY>	Designates replacement for the current key. String. Max length 64. Multi byte characters accepted.

© 2005 Blackboard Inc. Proprietary and Confidential

APPENDIX D—GLOSSARY OF TERMS

TERM	DEFINITION
Active	A *Blackboard Learning System* record in active state appears in the database and within the application.
API	Application Programming Interface. An API is code within an application that is revealed to allow programmers to create software that interacts with the application.
Available	A *Blackboard Learning System* active record that is available and can be managed and manipulated within the *Blackboard Learning System*. For example, a course that is available can be accessed by all users with a role in that course.
Batch operation	Processes several data records at once.
Batch UID	An attribute, or attributes, that are unique to each record within a data entity.
Change Key	Event Driven API persistence action. Changes the unique identifier for a person or group. The change will be reflected in any membership records as well.
Concrete Data Types and Abstract Data Types	Concrete data types, such as courses and users, appear within *Blackboard Learning System*. Abstract data types, such as group or person are IMS standards that relate to concrete, *Blackboard Learning System* data types.
Course	A data entity where each record consists of an individual course. Attributes identify details about the course, including: course name, course ID, and descriptive information.
Course Category	A data entity where each record consists of a course catalog category. Attributes include a category name, a category key called a mnemonic, and the availability of the category.
Course Category Membership	A data entity where each record connects a course to a course catalog category. Course Category Memberships are identified by a composite primary key composed of the Course ID and the category key.
Data attribute	A type of data that may be shared by the records in an entity. If viewing an entity as a table, an attribute is an individual column.
Data Ownership	Data attributes are owned by the institution information system or by *Blackboard Learning System*. If a data attribute is owned by *Blackboard Learning System*, than a difference between the feed file and the *Blackboard Learning System* database is ignored during a Snapshot operation.

© 2005 Blackboard Inc. Proprietary and Confidential

TERM	DEFINITION
Data Source Management	Identifies external data sets. All data that is integrated with *Blackboard Learning System* is tagged with an identifier that groups it with a logical set of data. The Data Source Management Tool can be used to not only identify, but to create a sophisticated management system to handle the interaction between the institution's information systems and *Blackboard Learning System*.
Deleted	A deleted record has been removed from *Blackboard Learning System*. The term "deleted" is used with the term "purged" interchangeably.
Disable	Event Driven API persistence action. Puts a record into disabled state.
Disabled	A *Blackboard Learning System* record in disabled state still exists in the *Blackboard Learning System* database but does not appear within the application.
Entity	A group of data records that share the same attributes. Each record is unique and identified by a key.
Field	A field within *Blackboard Learning System* is any item that accepts input from the user. Information entered into a field within the application is stored as an attribute within the database.
Group	IMS data entity that is represented by the Course and Organization entities in Blackboard.
Group Membership	IMS data entity that is represented by the Student Enrollment, Staff Assignment, and Organization Membership entities in *Blackboard Learning System*.
Insert	Event Driven API persistence action. Inserts a record into the *Blackboard Learning System* database.
Manual Mode	The feed file is processed and each record is either inserted or updated. If a record appears within *Blackboard Learning System* and not in the feed file, it is not changed. Manual mode is often used to explicitly perform an insert or update of a small number of records, unlike snapshot mode, where data is reconciled between *Blackboard Learning System* and the institution's information system.
Object	An object is an instance of a java class that contains both data attributes and methods for handling the data.
Organization	A data entity where each record consists of an individual organization. Attributes identify details about the organization, including: organization name, organization ID, and descriptive information.
Organization Category	A data entity where each record consists of an organization catalog category. Attributes include a category name, a category key called a mnemonic, and the availability of the category.

© 2005 Blackboard Inc. Proprietary and Confidential

TERM	DEFINITION
Organization Category Membership	A data entity where each record connects an organization to an organization catalog category. Organization Category Memberships are identified by a composite primary key composed of the Organization ID and the category key.
Organization Membership	A data entity where each record consists of a user's participation in a course. Organization Membership records are identified by a composite primary key composed of the user ID and the course ID.
Persister	Objects that control the data input into *Blackboard Learning System* as part of the Data Integration process.
Person	IMS data entity that is represented by the User entity in *Blackboard Learning System*.
Purged	A purged record has been removed from *Blackboard Learning System*. The term "purged" is used with the term "deleted" interchangeably.
Record	An item within an entity that contains the attributes of that entity. Viewing an entity as a table, a record would be an individual row within that table.
Remove	Event Driven API persistence action. Deletes records from the *Blackboard Learning System* database.
Remove Mode	The records in the feed file are permanently deleted from *Blackboard Learning System*. In addition, any associated records, such as enrollments or category links, are deleted as well.
Save	Event Driven API persistence action. Updates a record if it exists. If it does not, the record is inserted into the *Blackboard Learning System* database.
Smart Update	Synonymous with manual update, a smart update inserts or updates but does not change records that exist in the *Blackboard Learning System* database that are not in the feed file.
Snapshot	The Snapshot process takes a feed file generated from another of the institution's information systems and processes the data within *Blackboard Learning System*.
Snapshot Controller	The Snapshot Controller automates and manages the snapshot process after the Snapshot Generator creates a feed file. Like the Snapshot Generator, the institution must write the Snapshot Controller.
Snapshot Generator	Software, written by the institution, which generates flat files or XML files from another of the institution's information systems.
Snapshot Mode	The feed file is processed and each record is inserted, updated, or disabled.

© 2005 Blackboard Inc. Proprietary and Confidential

TERM	DEFINITION
Staff Assignment	A data entity where each record consists of a staff member's participation in a course. Staff Assignment records are identified by a composite primary key composed of the User Name and the course ID. The user's role in the course must be identified in the record.
Student Enrollment	A data entity where each record consists of a student's participation in a course. Student enrollment records are identified by a composite primary key composed of the User Name and the course ID.
Unavailable	A *Blackboard Learning System* active record that is unavailable will appear within the *Blackboard Learning System*, however, only users with certain roles can see it. For example, a course that is unavailable will not appear to students, but system administrators and instructors assigned to the course can view the course.
Update	Event Driven API persistence action. Updates a record in the *Blackboard Learning System* database.
User	A data entity where each record consists of an individual user. Attributes identify details about the user, including: user ID, user role, contact and other personal information.

© 2005 Blackboard Inc. Proprietary and Confidential

© 2005 Blackboard Inc. Proprietary and Confidential