Voyager® 8.2 BatchCat.dll Technical User's Guide

August 2012

ExLibris The bridge to knowledge

CONFIDENTIAL INFORMATION

The information herein is the property of Ex Libris Ltd. or its affiliates and any misuse or abuse will result in economic loss. DO NOT COPY UNLESS YOU HAVE BEEN GIVEN SPECIFIC WRITTEN AUTHORIZATION FROM EX LIBRIS LTD.

This document is provided for limited and restricted purposes in accordance with a binding contract with Ex Libris Ltd. or an affiliate. The information herein includes trade secrets and is confidential.

DISCLAIMER

The information in this document will be subject to periodic change and updating. Please confirm that you have the most current documentation. There are no warranties of any kind, express or implied, provided in this documentation, other than those expressly agreed upon in the applicable Ex Libris contract. This information is provided AS IS. Unless otherwise agreed, Ex Libris shall not be liable for any damages for use of this document, including, without limitation, consequential, punitive, indirect or direct damages.

Any references in this document to third-party material (including third-party Web sites) are provided for convenience only and do not in any manner serve as an endorsement of that third-party material or those Web sites. The third-party materials are not part of the materials for this Ex Libris product and Ex Libris has no liability for such materials.

TRADEMARKS

"Ex Libris," the Ex Libris bridge , Primo, Aleph, Alephino, Voyager, SFX, MetaLib, Verde, DigiTool, Preservation, URM, Voyager, ENCompass, Endeavor eZConnect, WebVoyage, Citation Server, LinkFinder and LinkFinder Plus, and other marks are trademarks or registered trademarks of Ex Libris Ltd. or its affiliates.

The absence of a name or logo in this list does not constitute a waiver of any and all intellectual property rights that Ex Libris Ltd. or its affiliates have established in any of its products, features, or service names or logos.

Trademarks of various third-party products, which may include the following, are referenced in this documentation. Ex Libris does not claim any rights in these trademarks. Use of these marks does not imply endorsement by Ex Libris of these third-party products, or endorsement by these third parties of Ex Libris products.

Oracle is a registered trademark of Oracle Corporation.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd.

Microsoft, the Microsoft logo, MS, MS-DOS, Microsoft PowerPoint, Visual Basic, Visual C++, Win32, Microsoft Windows, the Windows logo, Microsoft Notepad, Microsoft Windows Explorer, Microsoft Internet Explorer, and Windows NT are registered trademarks and ActiveX is a trademark of the Microsoft Corporation in the United States and/or other countries.

Unicode and the Unicode logo are registered trademarks of Unicode, Inc.

Google is a registered trademark of Google, Inc.

Document released: August 2012

Web address: http://www.exlibrisgroup.com

Contents

Contents

Contents

Contents

Contents

Contents

About This Document

Purpose

This document provides instructions for working with Voyager dynamic link libraries.

Intended Audience

This document is intended for technical staff supporting Voyager systems.

Reason for Reissue

This user's guide incorporates and is being reissued for the following reasons:

- Addition of a Note on page 2-29
- Addition of return code 212 to Table 2-17 on page 2-29

Document Summary

This document consists of the following:

Chapter 1	"Getting Started Chapter 1 provides general information about working with BatchCat.dll.
Chapter 2	Voyager Dynamic Link Libraries Chapter 2 provides specific information about using the Voyager dynamic link libraries using BatchCat.dll.
Index	The Index is an alphabetical, detailed cross-reference of topics.

Conventions Used in This Document

The following conventions are used throughout this document:

- Names of commands, variables, stanzas, files, and paths (such as `/dev/tmp`), as well as selectors and typed user input, are displayed in `constant width` type.
- Commands or other keyboard input that must be typed exactly as presented are displayed in **`constant width bold`** type.
- Commands or other keyboard input that must be supplied by the user are displayed in ***`constant width bold italic`*** type.
- System-generated responses such as error messages are displayed in `constant width` type.
- Variable *portions* of system-generated responses are displayed in *`constant width italic`* type.
- Keyboard commands (such as **Ctrl** and **Enter**) are displayed in **bold**.
- Required keyboard input such as "Enter **`vi`**" is displayed in **`constant width bold`** type.
- Place holders for variable portions of user-defined input such as **`ls -l`** ***`filename`*** are displayed in ***`italicized constant width bold`*** type.
- The names of menus or status display pages and required selections from menus or status display pages such as "From the **Applications** drop-down menu, select **System-wide**," are displayed in **bold** type.
- Object names on a window's interface, such as the **Description** field, the **OK** button, and the **Metadata** tab, are displayed in **bold** type.
- The titles of documents such as *Curator Web Client User's Guide* are displayed in *italic* type.

- Caution, and important notices are displayed with a distinctive label such as the following:

NOTE:
Extra information pertinent to the topic.

IMPORTANT:
Information you should consider before making a decision or configuration.

CAUTION:
Information you must consider before making a decision, due to potential loss of data or system malfunction involved.

TIP:
Helpful hints you might want to consider before making a decision.

RECOMMENDED:
Preferred course of action.

OPTIONAL:
Indicates course of action which is not required, but may be taken to suit your library's preferences or requirements.

Document Reproduction/Photocopying

Photocopying the documentation is allowed under your contract with Ex Libris (USA) Inc. It is stated below:

> All documentation is subject to U.S. copyright protection. CUSTOMER may copy the printed documentation only in reasonable quantities to aid the employees in their use of the SOFTWARE. Limited portions of documentation, relating only to the public access catalog, may be copied for use in patron instruction.

Comment on This Document

To provide feedback regarding this document, use the Ex Libris eService or send your comments in an e-mail message to docmanager@exlibrisgroup.com.

Getting Started

1

Getting Started

1

Introduction

This chapter describes information and skills required to work with Voyager dynamic link libraries.

Purpose of this Chapter

This chapter's purpose is to provide information about the skills and knowledge required to use BatchCat.dll to process cataloging records in batch mode versus one record at a time.

Prerequisite Skills and Knowledge

To use this document effectively, you should have a general understanding of the Voyager Cataloging client. You should also be familiar with the items listed below.

- Microsoft Windows-based applications
- Microsoft Visual Basic
- At least one programming language that can access ActiveX Dynamic Link Libraries.
- UNIX operating system commands and file system
- Oracle Database system including SQL and SQL*Plus

- At least one UNIX-based text editor such as `ed` or `vi`
- Local procedures

Voyager Dynamic Link Libraries

2

Voyager Dynamic Link Libraries

2

Introduction

BatchCat.dll provides an alternative to manipulating MARC and item records one at a time using Voyager's Cataloging module.

It allows you to add, update, and delete bibliographic, holdings, authority, and item records in batches without the use of the Cataloging client.

The following three components are involved in this process:

- External (non-Voyager) program
- BatchCat.dll
- Voyager database

The External Program

The actual adding, deleting, or updating of MARC and item records occurs in an external program that you code or acquire by some other means.

Since Ex Libris (USA) Inc. is not responsible for creating or distributing the external program, this documentation only refers to it in its capacity to work with BatchCat.dll. The dynamics of the external program such as how it communicates with Voyager and how it obtains records from Voyager are not addressed.

The external program obtains records from your Voyager database and provides a means by which you can manipulate them. You can create the program using any programming language provided that it can access ActiveX Dynamic Linked Libraries (DLLs). The examples used throughout this documentation pertain to Microsoft's Visual Basic.

NOTE:
If you choose to create an external program with Visual Basic, BatchCat.dll displays in the list of Visual Basic's references that can be added to a project as a Voyager Batch Cataloging Utility.

How the External Program Works with BatchCat.dll

Like any DLL, BatchCat.dll is not a program but rather a set of functions that need to be called by another program in order to operate.

BatchCat.dll is loaded into and then referenced by the external program you create. Once encapsulated in the program, BatchCat.dll serves to send added, updated, or deleted records one at a time to the Voyager database. This facilitates the processing of records in batches. It allows you to bring many records at once into the external program for manipulation.

Records that you work with may be new additions to the Voyager database. That is, they may have been imported into the external program from a bibliographic utility and then tailored to meet your needs.

Or records may have originated from the Voyager database. That is, they were edited in the external program to accommodate new Cataloging policies or changes in holdings.

For BatchCat.dll to send records to the Voyager database, it contains routines you must implement.

About ClassBatchCat

There is a class defined inside BatchCat.dll called ClassBatchCat. ClassBatchCat allows you to communicate with BatchCat.dll.

You must create an instance of ClassBatchCat that invokes its public properties, objects, and functions such as in the following statement where xxx = the instance name.

Example:

```
Private <xxx> As New ClassBatchCat
```

Public Properties, Objects, and Functions

ClassBatchCat's public properties, objects, and functions are listed in Table 2-1.

Table 2-1. ClassBatchCat - Public Properties, Objects, & Functions

Properties	Objects	Functions
RecordIDAdded	cItem	AddAuthorityRecord
		AddBibRecord
		AddHoldingRecord
		AddItemBarcode
		AddItemData
		AddItemStatus
		Connect
		DeleteAuthorityRecord
		DeleteBibRecord
		DeleteHoldingRecord
		DeleteItemRecord
		DeleteItemStatus
		UpdateAuthorityRecord
		UpdateBibRecord
		UpdateHoldingRecord
		UpdateItemData

Public Property of ClassBatchCat - RecordIDAdded

RecordIDAdded is the only public property of ClassBatchCat. If you successfully add a record, RecordIDAdded contains the Voyager ID that was added to the Voyager database.

Public Object of ClassBatchCat - cItem

cItem is the only public object of ClassBatchCat. It is an instantiated object that is automatically created for you when ClassBatchCat is created. cItem holds the values for adding and updating item records.

cItem Variables - Overview

There are 17 variables for the cItem object. The first four are required and the remaining 13 are optional. Variables are expected to contain valid values before invoking the methods that use the cItem object.

Setting Variables. The required and optional variables of the cItem object can be set in the following manner where <xxx> = the name of the instance for ClassBatchCat.

Example:

```
<xxx>.cItem.MfhdID = 1234
<xxx>.cItem.CatLocationID = 1
<xxx>.citem.FreeText = <Insert verbiage...>
<xxx>.cItem.SequenceNewItemsAtTop = True
```

Default Values. All of cItem's variables are reset to their default values when the object is created or cleared. Default values are included in Table 2-2 and Table 2-3.

cItem Variables - Required

The four variables listed in alphabetical order in Table 2-2 are required for the ClassBatchCat's cItem object.

NOTE:
Each variable is of a specific type (Long, String, Boolean, or Integer), applies to a specific function (adds, updates, or both), is mandatory for either adds, updates,

or both, has a particular default value, and indicates something specific such as a Voyager ID for a holding record.

Table 2-2. cItem Object - Required Variables

Name	Type	Specifics	Description and Miscellaneous Information
HoldingID	Long	Mandatory for adds No default (invalid if zero)	Indicates the Voyager ID for a holding record being added.
ItemID	Long	Applies to updates only Mandatory for updates No default (invalid if zero)	Indicates the Voyager ID of an item record being updated. Used only for updates because for adds, the system will generate a new Item ID upon successful completion of the add and will return the generated Item ID in the "RecordIDAdded" property of ClassBatchCat. Thus, if an Item ID is passed on an item add, it will be ignored.
ItemTypeID	Long	Applies to adds and updates Mandatory for adds and updates No default (invalid if zero)	Indicates the valid item type ID from the item_type table. If there is no change for updates, use the item_type_id table.
Perm LocationID	Long	Applies to adds and updates Mandatory for updates No default (invalid if zero)	Indicates the valid Voyager location ID of an item record being added or updated. If item records are updated and no change is made to the PermLocationID, then the perm_location table is used. For adds, zero tells the Voyager Server to use the holding location; for updates, zero will cause an error to occur.

cItem Variables - Optional

The 13 variables listed in alphabetical order in Table 2-3 are optional for ClassBatchCat's cItem object. Each variable has a default that can be used for adds but must be overridden for updates with existing or new values. Leaving the default for updates changes the current item settings to the defaults.

NOTE:
Each optional variable is of a specific type (Long, String, Boolean, or Integer), has a particular default value, indicates something specific such as Voyager location ID, and receives its data from a particular table if there are no changes made. In addition, some variables have a maximum on the number of characters allowed.

Table 2-3. cItem Object - Optional Variables

Name	Type	Specifics	Description and Miscellaneous Information
AddItemTo Top	Boolean	Default = False	Only used for adds. True = the new item is added at the top of the current sequence False = the new item is added at the bottom of the current sequence.
Caption	String	Default = zero length string Max 256 characters (excess truncated)	If no change for updates, use caption in the mfhd_item table.
Chron	String	Default = zero length string Max 80 characters (excess truncated)	If no change for updates, use caption in the mfhd_item table.
CopyNumber	Integer	Default = 0	Indicates a copy number between 0 and 32767 If no change for updates, use copy_number in item table.

Name	Type	Specifics	Description and Miscellaneous Information
Enumeration	String	Default = zero length string Max 80 characters (excess truncated)	If no change for updates, use item_enum in the mfhd_item table.
FreeText	String	Default = zero length string Max 256 characters (excess truncated)	If no change for updates, use freetext in the mfhd_item table.
MediaTypeID	Long	Default = no media type	Indicates a media type ID in the media_type table. If no change for updates, use media_type_id from item table.
PieceCount	Integer	Default = 1	Indicates a piece count number between 1 and 9999. If no change for updates, use pieces from the item table.
Price	String	Default = 0	If setting this variable, use the correct number of decimal places and do not use currency or separator symbols. If no price is supplied, then the default (“0”) is assumed. If no change for updates, use price in the item table.
SpineLabel	String	Default = zero length string Max 25 characters (excess truncated)	If no change for updates, use spine_label in the item table.
Temp LocationID	Long	Default = no temp location	Indicates the valid Voyager location ID in the location table. If no change for updates, use temp_location in item table.

Name	Type	Specifics	Description and Miscellaneous Information
TempTy-peID	Long	Default = no temp item type	Indicates the item type ID from the item_type table. If no change for updates, use media_type_id from item table.
Year	String	Default = zero length string Max 80 charac-ters (excess trun-cated)	If no change for updates, use year in the mfhd_item table.

Public Functions of ClassBatchCat

There are 16 public functions of ClassBatchCat each performing a specific task for a particular record type. The tasks and record types are indicated by the name of the function such as the AddBibRecord function used for adding a bibliographic record. Public functions have required parameters, perform some sort of validation, and return something to you once a task is performed like return codes.

Return Codes and Server Errors

Return codes for each function are generated by BatchCat.dll unless specified as server errors. Server errors occur when the data sent to the DLL is correct; but for some reason, the server cannot perform a function such as data is being used by another user or a user does not have correct security. BatchCat.dll errors occur when the data sent to the DLL is incorrect or missing.

Invoking Public Functions

The public functions of ClassBatchCat are invoked as in the following statement for calling the Connect function where <xxx> = the name of the instance for ClassBatchCat.

Example:

```
<xxx>.Connect (App, JohnDoe, John1234)
```

ClassBatchCat Public Functions Descriptions

The remainder of this technical guide contains detailed descriptions of ClassBatchCat's 16 public functions. It is intended to be used as a reference tool for invoking specific functions. Hence, functions are presented in alphabetical order.

Each ClassBatchCat public function description is divided into the following sections.

- Required Parameters
- Validation
- Return Codes

Remember that the name of each function connotes the task it performs and the type of record to which it pertains.

AddAuthorityRecord Function

The following describes the AddAuthorityRecord function.

Required Parameters

The following are required parameters for the AddAuthorityRecord function.

MARCRecord

Type: String.

Values: A valid MARC Authority record.

CatLocationID

Type: Long.

Values: A valid Cataloging Location ID.

Validation

The following is validated with the AddAuthority function.

1. Validates the CatLocationID.
2. Validates that the record is a true MARC record.

3. Validates that the record has a leader field.
4. Validates that the record is flagged as an authority record.
5. Validates that the record has at least one 1xx field.
6. Validates that the 008 field exists and is in the correct format.

Return Codes

The following return codes are provided for the AddAuthorityRecord function.

RecordIDAdded

Type: Long.

Values: Public property of the ClassBatchCat object set to the Voyager record ID if the add was successful. Set to zero otherwise.

AddAuthorityReturnCode

Type: Integer.

Values: See Table 2-4.

Table 2-4. AddAuthorityReturnCode Values

0	aaSuccess
1	aaUnknownError
2	aaNotAValidRecord
3	aaInvalidRecordType
5	aaCouldNotInsert001
10	aaCouldNotAddRecor
15	aaInvalidLeaderLength
16	aaNoLeaderField
21	aaRecordEditError
24	aaInvalidCatalogingLocationID
31	aaCouldNotAssembleRecord
32	aaCouldNotCreateNewID
33	aaInvalid008Length
34	aaRecordHasNo008Data

Table 2-4. AddAuthorityReturnCode Values

36	aaRecordStructureIsInvalid
203	aaReadOnly (Server Error)

AddBibRecord Function

The following describes the AddBibRecord function.

Required Parameters

The following are required parameters for the AddBibRecord function.

MARCRecord

Type: String.

Values: Valid MARC 21 bibliographic record (only Unicode™ records).

LibraryID

Type: Long.

Values: Valid owning library ID.

CatLocationID

Type: Long.

Values: Valid Cataloging location ID.

OpacSuppress

Type: Boolean.

Values: True/False.

Validation

The following is validated with the AddBibRecord function.

1. Validates the Cataloging location ID.
2. Validates the library ID.
3. Validates that the record is a true MARC bibliographic record.
4. Validates that a title exists in the 245 field.

5. Validates that the 008 field exists and is the correct length.

Return Codes

The following return codes are provided for the AddBibRecord function.

RecordIDAdded

Type: Long.

Values: Public property of the ClassBatchCat object set to the Voyager record ID if the add was successful. Set to zero otherwise.

AddBibReturnCode

Type: Integer.

Values: See Table 2-5.

Table 2-5. AddBibReturnCode Values

0	abSuccess
1	abUnknownError
2	abNotAValidRecord
3	abInvalidRecordType
4	abRecordHasNoTitle
5	abCouldNotInsert001
10	abCouldNotAddRecord
15	abInvalidLeaderLength
16	abNoLeaderField
21	abRecordEditError
22	abInvalidLibraryID
24	abInvalidCatalogingLocationID
31	abCouldNotAssembleRecord
32	abCouldNotCreateNewID
33	abInvalid008Length
34	abRecordHasNo008Data
36	abRecordStructureIsInvalid
203	abRead Only (Server Error)

AddHoldingRecord Function

The following describes the AddHoldingRecord function.

Required Parameters

The following are required parameters for the AddHoldingRecord function.

MARCRecord

Type: String.

Values: A valid MARC holding record.

RelatedRecordID

Type: Long.

Values: The linked bib ID (stored in the 004 field upon success).

CatLocationID

Type: Long.

Values: A valid Cataloging location ID.

OpacSuppress

Type: Boolean.

Values: True / False.

HoldLocationID

Type: Long.

Values: A valid holding location ID such as the location ID that matches the location code in the 852 subfield b.

Validation

The following is validated with the AddHoldingRecord function.

1. Validates the CatLocationID.
2. Validates that the record is a true MARC record.
3. Validates the data in 852 subfield b.

4. Validates that the passed RelatedRecordID exists.
5. Validates that the 008 field exists and is in the correct format.

Return Codes

The following return codes are provided for the AddHoldingRecord function.

RecordIDAdded

Type: Long.

Values: Public property of the ClassBatchCat object set to the Voyager record ID if the add was successful. Set to zero otherwise.

AddHoldingReturnCode

Type: Integer

Values: See Table 2-6.

Table 2-6. AddHoldingReturnCode Values

0	ahSuccess
1	ahUnknownError
2	ahNotAValidRecord
3	ahInvalidRecordType
5	ahCouldNotInsert001
10	ahCouldNotAddRecord
15	ahInvalidLeaderLength
16	ahNoLeaderField
20	ahCouldNotAddRelatedID
21	ahRecordEditError
23	ahInvalidLocationCode
24	ahInvalidCatalogingLocationID
31	ahCouldNotAssembleRecord
32	ahCouldNotCreateNewID
33	ahInvalid008Length
34	ahRecordHasNo008Data
35	ahLinkedIdError

Table 2-6. AddHoldingReturnCode Values

36	ahRecordStructureIsInvalid
37	ahRecordHasNo852Field
38	ah852FieldHasNoSubfieldB
39	ah852SubfieldBHasNoData
203	ahReadOnly (Server error)

AddItemBarcode Function

The following describes the AddItemBarcode function.

Required Parameters

ItemID

Type: Long.

Values: The Item Id for the new barcode.

Barcode

Type: String.

Values: The barcode being added.

Validation

The following is validated with the AddItemBarcode function.

1. Validates that the ItemID exists.
2. Validates that the Barcode is 25 alpha-numeric characters long (excess truncated)

Return Codes

The following return codes are provided for the AddItemBarcode function.

AddItemBarCodeReturnCode

Type: Integer

Values: See Table 2-7.

Table 2-7. AddItemBarCodeReturnCode Values

0	ibSuccess
1	ibUnknownError
80	ibInvalidItemId
81	ibInvalidBarCode
82	ibCouldNotUpdateBarcode

AddItemData Function

The following describes the AddItemData function.

Required Parameters

The following are required parameters for the AddItemData function.

CatLocationID

Type: Long.

Values: A valid Cataloging Location ID.

NOTE:
Other parameters need to be set in ClassBatchCat's cItem property. See "Public Object of ClassBatchCat - cItem" on page 2-4 for more information.

Validation

The following is validated with the AddItemData function.

1. Validates the CatLocationID.
2. Validates the cItem.HoldingID value.
3. Validates the cItem.PermLocationID value.
4. Validates the cItem.ItemType value.
5. Validates the cItem.TempTypeID value.
6. Validates the cItem.TempLocation value.
7. Validates the cItem.MediaTypeID value.
8. Validates the cItem.CopyNumber is between 0 and 32767 (inclusive).

9. Validates the cItem.PieceCount is between 0 and 9999 (inclusive).
10. Validates the cItem.Price is numeric and not negative.

Return Codes

The following return codes are provided for the AddItemData function.

RecordIDAdded

Type: Long.

Values: Public property of the ClassBatchCat object set to the Voyager record ID if the add was successful. Set to zero otherwise.

AddItemReturnCode

Type: Integer.

Values: See Table 2-8.

Table 2-8. AddItemReturnCode Values

0	aiSuccess
1	aiUnknownError
24	aiInvalidCatalogingLocationID
50	aiCouldNotAddItemRecord
52	aiInvalidPrice
55	aiInvalidHoldingID
56	aiInvalidPermLocationID
57	aiInvalidTempLocationID
58	aiInvalidMediaTypeID
59	aiInvalidPermTypeCode
60	aiInvalidTempTypeCode

NOTE:
An item status of Not Charged (item status code = 1) is automatically assigned to a newly added item record.

AddItemStatus Function

The following describes the AddItemStatus function.

Required Parameters

The following are required parameters for the AddItemStatus function.

ItemID

Type: Long.

Values: The Item Id for the new status.

ItemStatus

Type: Long.

Values: The item status code being added (item_status_type in item_status_type table).

Validation

The following is validated with the AddItemStatus function.

1. Validates that the ItemID exists.
2. Validates the item status.

Return Codes

The following return codes are provided for the AddItemStatus function.

AddItemStatusReturnCode

Type: Integer.

Values: See Table 2-9.

Table 2-9. AddItemStatusReturnCode Values

0	aisSuccess
1	aisUnknownError
80	aisInvalidItemId
90	aisItemBlocked
91	aisItemStatusUpdateFailed

Table 2-9. AddItemStatusReturnCode Values

92	aisNotAValidItemStatus

Connect Function

The following describes the Connect function.

Required Parameters

The following are required parameters for the Connect function.

AppPath

Type: String.

Values: Path to the `voyager.ini` file.

UserName

Type: String.

Values: User's userID.

Password

Type: String.

Values: User's password.

Validation

The following is validated with the Connect function.

1. Validates that the AppPath parameter is that of the voyager.ini file.
2. Validates the username and password.

Return Codes

The following return codes are provided for the Connect function.

ConnectReturnCodes

Type: Integer.

Values: See Table 2-10.

Table 2-10. ConnectReturnCodes Values

0	crSuccess
1	crUnknownError
26	crCannotConnect
27	crInvalidUserNameOrPassword
28	crCannotGetCorrectSecurity
29	crCouldNotGetItemStatuses

NOTE:
If the connect fails, you may receive error messages. When there is a connect failure, the same ConnectReturnCode value is sent for all other attempted functions. BatchCat.dll uses the Voyager.dll Connect function to connect to the Cataloging server that services all the database requests. Messages typically address either errors in the `voyager.ini` file or larger system errors such as Oracle being down.

DeleteAuthorityRecord Function

The following describes the DeleteAuthorityRecord function.

Required Parameters

The following are required parameters for the DeleteAuthorityRecord function.

RecordID

Type: Long.

Values: A valid Authority record ID.

Validation

The following is validated with the DeleteAuthorityRecord function.

- Validates that the RecordID exists.

Return Codes

The following return codes are provided for the DeleteAuthorityRecord function.

DeleteAuthorityReturnCode

Type: Integer.

Values: See Table 2-11.

Table 2-11. DeleteAuthorityReturnCode Values

0	daSuccess
1	daUnknownError
13	daInvalidRecordID
203	daReadOnly (Server error)

DeleteBibRecord Function

The following describes the DeleteBibRecord function.

Required Parameters

The following are required parameters for the DeleteBibRecord function.

RecordID

Type: Long.

Values: A valid bibliographic record ID.

Validation

The following is validated with the DeleteBibRecord function.

- Validates that the RecordID exists.

Return Codes

The following return codes are provided for the DeleteBibRecord function.

DeleteBibRecordCode

Type: Integer.

Values: See Table 2-12.

Table 2-12. DeleteBibRecordCode Values

0	dbSuccess
1	dbUnknownError
13	dbInvalidRecordID
102	dbMfhdsAttached (Server error)
103	dbLineItemsAttached (Server error)
105	dbOpacRequestAttached (Server error)
107	dbHoldRecallAttached (Server error)
203	dbReadOnly (Server error)

DeleteHoldingRecord Function

The following describes the DeleteHoldingRecord function.

Required Parameters

The following are the required parameters for the DeleteHoldingRecord function.

RecordID

Type: Long.

Values: A valid holding record ID.

Validation

The following is validated with the DeleteHoldingRecord function.

- Validates that the RecordID exists.

Return Codes

The following return codes are provided for the DeleteHoldingRecord function.

DeleteHoldingReturnCode

Type: Integer.

Values: See Table 2-13.

Table 2-13. DeleteHoldingReturnCode Values

0	dhSuccess
1	dhUnknownError
13	dhInvalidRecordID
100	dhItemsRecordsAttached (Server error)
101	dhLineItemsCopyAttached (Server error)
104	dhEItemsAttached (Server error)
105	dhOpacRequestAttached (Server error)
106	dhCallslipAttached (Server error)
107	dhHoldRecallAttached (Server error)
203	dhReadOnly (Server error)

DeleteItemRecord Function

The following describes the DeleteItemRecord function.

Required Parameters

The following are required parameters for the DeleteItemRecord function.

RecordID

Type: Long.

Values: A valid Item record ID.

Validation

The following is validated with the DeleteItemRecord function.

- Validates that the RecordID exists.

Return Codes

The following return codes are provided for the DeleteItemRecord function.

DeleteItemReturnCode

Type: Integer.

Values: See Table 2-14.

Table 2-14. DeleteItemReturnCode Values

0	diSuccess
1	diUnknownError
13	diInvalidRecordID
61	diItemChargedToPatron
62	diItemHasExceptionLogged
63	diItemHasFineOrFee
64	diItemIsOnReserve
65	diItemHasAShortLoanRequest
66	diItemIsADistributionItemOnOrder
67	diItemIsCurrentlyScheduled
68	diItemOnHoldOrAwaitingARequestCancellation Notice
69	diItemHasBeenQueuedForACallslip
70	diItemIsInRemoteStorage
71	diItemMayBeNeededToFulfillABooking
72	diItemIsADistributionItem
105	diOpacRequestAttached (Server error)
106	diCallslipAttached (Server error)
107	diHoldRecallAttached (Server error)
108	diMediaItemAttached (Server error)

DeleteItemStatus Function

The following describes the DeleteItemStatus function.

Required Parameters

The following are required parameters for the DeleteItemStatus function.

ItemID

Type: Long.

Values: The Item Id of the deleted status.

ItemStatus

Type: Long.

Values: The Item Status code being removed (item_status_type in item_status_type table).

Validation

The following is validated with the DeleteItemStatus function.

1. Validates that the itemID exists.
2. Validates the item status.

Return Codes

The following return codes are provided for the DeleteItemStatus function.

DeleteItemStatusReturnCode

Type: Integer.

Values: See Table 2-15.

Table 2-15. DeleteItemStatusReturnCode Values

0	disSuccess
1	disUnknownError
80	disInvalidItemId
90	disItemBlocked
91	disItemStatusUpdateFailed
92	disNotAValidItemStatus

ItemBoundWith Function

This API binds an item record and its holdings record to a bibliographic record. This is a wrapper that takes the parameters passed into it and forwards them to a server API to do the linking.

Required Parameters

The following are the required parameters for the ItemBoundWith function.

ItemID

Type: Long

Values: The ID of the item to be bound with

HoldingID

Type: Long

Values: The ID of the holdings record of the above item

BibID

Type: Long

Values: The ID of the bibliographic record the item and holdings are to be bound to

CatLocationID

Type: Long

Values: The cataloging location ID

Validation

The input data is validated per the steps below. Failed verifications return valid return codes identifying what condition failed and result in the relinking operation being skipped.

1. Verify that the item, holdings, bibliographic, and location data passed in are valid with a server API call.
2. Get the current bound with status of the item by calling a server API which returns one bibliographic ID if not bound with or multiple bibliographic IDs if bound with.
3. Check the new bibliographic ID per the Cataloging client to verify that it is different from the bibliographic IDs the item is currently linked to.
4. If the Ignore Hold Ownership flag is OFF, use server APIs to verify that the library IDs are the same for holdings and the new bibliographic record.

   The Ignore Hold Ownership flag is located in the `cat_profile` table.
5. Call the bound with server API.
6. With a successful link, update the holdings record.

a. Use the server API to get the MARC data.
b. Add a 014 tag containing the new bibliographic ID.
c. Save the modified holdings with a server API call.

Return Codes

The following are the return codes provided for ItemBoundWith.

ItemBoundWithReturnCode

Type: Long

Values: See Table 2-16

Table 2-16. ItemBoundWithReturnCode Values

0	ibwSuccess
1	ibwUnknownError
10	ibwCouldNotAddRecord
13	ibwInvalidRecordID
24	ibwInvalidLocationID
36	ibwRecordStructureIsInvalid
55	ibwInvalidHoldingID
80	ibwInvalidItemID
93	ibwUnableToLinkRecord
95	ibwLinkedRecsDifLibs
96	ibwBibUnable2Get
97	ibwHoldUnable2Get
98	ibwUnable2GetBoundInformation
99	ibwAlreadyLinked

RelinkHoldingRecord Function

This API links a holdings record to a different bibliographic record. This is a wrapper that takes the parameters passed into it and forwards them to the server API to do the actual linking.

Required Parameters

The following are the required parameters for the RelinkHoldingRecord function.

HoldingID

Type: Long

Values: The ID of the holdings record to be relinked

BibSourceID

Type: Long

Values: The ID of the bibliographic record that the holdings record was originally linked to

BibID

Type: Long

Values: The ID of the bibliographic record that the holdings record is relinked to

CatLocationID

Type: Long

Values: The cataloging location ID

Validation

The input data is validated per the steps below. Failed verifications return valid return codes identifying what condition failed and result in the relinking operation being skipped.

1. Compare the old bibliographic ID and the new bibliographic ID.

   They must be different.

2. Verify that the holdings, bibliographic records, and the location passed in are valid.
3. If the Ignore Hold Ownership flag is OFF, use the server APIs to verify that the library IDs are the same for the holdings and new bibliographic records.

   The Ignore Hold Ownership flag is located in the `cat_profile` table.

4. Call the relink server API.
5. With a successful link, update the holdings record.

a. Use the server API to get the MARC data.

b. Loop over all 014 tags looking for a reference to the old bibliographic record and if found:

   i. Remove the 014 tag.

   ii. Save the modified holdings record with the server API.

NOTE:
If the holdings record is currently linked to multiple bibliographic records, the holdings record is unlinked only from the old bibliographic ID specified and relinked to the new bibliographic ID specified leaving the other potential links intact.

NOTE:
If one or more bibliographic records (identified for relinking) are linked to any purchase order line items, the request to relink the holdings record is blocked and return coded 212 is sent.

Return Codes

The following are the return codes provided for RelinkHoldingRecord.

RelinkHoldingRecordReturnCode

Type: Long

Values: See Table 2-17

Table 2-17. RelinkHoldingRecordReturnCode Values

0	rhrSuccess
1	rhrUnknownError
10	rhrCouldNotAddRecord
13	rhrInvalidRecordID
19	rhrInvalidRelatedRecordID
24	rhrInvalidLocationID
55	rhrInvalidHoldingID
93	rhrUnableToLinkRecord
94	rhrDupIdsPassed
95	rhrLinkedRecsDifLibs
96	rhrBibUnable2Get

Table 2-17. RelinkHoldingRecordReturnCode Values

97	rhrHoldUnable2Get
212	rhrHoldIsLinkedToPO

RelinkItemRecord Function

This API links an item record to a different holdings record. This is a wrapper that takes the parameters passed into it and forwards them to a server API to do the actual linking.

Required Parameters

The following are the required parameters for the RelinkItemRecord function.

ItemID

Type: Long

Values: The ID of the item to be relinked

HoldingSourceID

Type: Long

Values: The ID of the holdings record that the item record was originally linked to

HoldingID

Type: Long

Values: The ID of the holdings record that the item is relinked to

StopIfNewHoldBoundWith

Type: Boolean

Values: A flag explicitly set to either TRUE or FALSE to control the action taken if the HoldingID above is already linked to multiple bibliographic records

StopIfOldHoldBoundWith

Type: Boolean

Values: A flag explicitly set to either TRUE or FALSE to control the action taken if the HoldingSourceID above is already linked to multiple bibliographic records

Validation

The input data is validated per the steps below. Failed verifications return valid return codes identifying what condition failed and result in the relinking operation being skipped.

The Cataloging client normally prompts when either the selected holdings record is linked to multiple bibliographic records or the the item is currently linked to a holdings record linked to multiple bibliographic records. One, both, or none of these warnings may display but the check is always carried out. When a warning is displayed, the client gives the option to continue with the new linking or cancel it.

IMPORTANT:
BatchCat cannot offer this interaction.

In order to emulate this functionality, BatchCat needs to get two extra parameters to direct it to either stop the operation or proceed with it depending on the current linking results.

If the flag is set to stop (TRUE), BatchCat checks for the condition to which the flag applies and stops if the condition is found to be true or proceed if the condition is found to be false. In the case of the operation being stopped, the appropriate return code is returned.

If the flag is set to continue or ignore (FALSE), BatchCat is not checked for the condition to which the flag applies and proceeds as if the condition, if checked and found true, had been found to be false.

1. Compare the old holdings ID and the new holdings ID.

   They must be different.

2. Verify that the item and holdings passed in are valid.
3. Check the flags described above and carry out the `bound_with` checks if appropriate.
4. Call the relink server API function.

Return Codes

The following are the return codes provided for RelinkItemRecord.

RelinkItemRecordReturnCode

Type: Long

Values: See Table 2-18

Table 2-18. RelinkItemRecordReturnCode Values

0	rirSuccess
1	rirUnknownError
19	rirInvalidRelatedRecordID
55	rirInvalidHoldingID
80	rirInvalidItemID
83	rirHoldingLinked2Multiple
84	rirItemLinked2Multiple
93	rirUnableToLinkRecord
94	rirDupIdsPassed

UpdateAuthorityRecord Function

The following describes the UpdateAuthorityRecord function.

Required Parameters

The following are required parameters for the UpdateAuthorityRecord function.

RecordID

Type: Long.

Values: A valid Authority record ID.

MARCRecord

Type: String.

Values: A valid MARC Authority record.

DateTime

Type: VBDate.

Values: The date/time stamp indicating when the record was last updated. If there is no update date/time, then the create date/time is used. This is the update_date (or create_date) from the auth_master table.

CatLocationID

Type: Long.

Values: A valid Cataloging Location ID.

Validation

The following is validated with the UpdateAuthorityRecord function.

1. Validates the CatLocationID.
2. Validates that the record is a true MARC authority record.
3. Validates the RecordID.

Return Codes

The following return codes are provided for the UpdateAuthorityRecord function.

UpdateAuthorityReturnCode

Type: Integer.

Values: See Table 2-19.

Table 2-19. UpdateAuthorityReturnCode Values

0	uaSuccess
1	uaUnknownError
2	uaNotAValidRecord
3	uaInvalidRecordType
7	uaRecordCanOnlyHaveOne001Field
8	uaRecordNeedsA001Field
13	uaInvalidRecordID
15	uaInvalidLeaderLength
16	uaNoLeaderField

Table 2-19. UpdateAuthorityReturnCode Values

21	uaRecordEditError
24	uaInvalidCatalogingLocationID
31	uaCouldNotAssembleRecord
33	uaInvalid008Length
34	uaRecordHasNo008Data
36	uaRecordStructureIsInvalid
41	uaDateTimeStampsDoNotMatch
42	uaCouldNotUpdateRecord
203	uaReadOnly (Server Error)

UpdateBibRecord Function

The following describes the UpdateBibRecord function.

Required Parameters

The following are required parameters for the UpdateBibRecord function.

RecordID

Type: Long.

Values: A valid bibliographic record ID.

MARCRecord

Type: String.

Values: A valid bibliographic record.

DateTime

Type: VBDate.

Values: The date/time stamp indicating when the record was last updated. If there is no update date/time, then the create date/time is used. This is the update_date (or create_date) from the bib_master table.

LibraryID

Type: Long.

Values: A valid owning library ID.

CatLocationID

Type: Long.

Values: A valid Cataloging location ID.

OpacSuppress

Type: Boolean.

Values: True / False.

Validation

The following is validated with the UpdateBibRecord function.

1. Validates the CatLocationID.
2. Validates the libraryID.
3. Validates that the record is a true MARC bibliographic record.
4. Validates that a title exists in that 245 field.
5. Validates the RecordID.

Return Codes

The following return codes are provided with the UpdateBibRecord function.

UpdateBibReturnCode

Type: Integer.

Values: See Table 2-20.

Table 2-20. UpdateBibReturnCode Values

0	ubSuccess
1	ubUnknownError
2	ubNotAValidRecord
3	ubInvalidRecordType

Table 2-20. UpdateBibReturnCode Values

4	ubRecordHasNoTitle
7	ubRecordCanOnlyHaveOne001Field
8	ubRecordNeedsA001Field
13	ubInvalidRecordID
15	ubInvalidLeaderLength
16	ubNoLeaderField
21	ubRecordEditError
22	ubInvalidLibraryID
24	ubInvalidCatalogingLocationID
31	ubCouldNotAssembleRecord
33	ubInvalid008Length
34	ubRecordHasNo008Data
36	ubRecordStructureIsInvalid
41	ubDateTimeStampsDoNotMatch
42	ubCouldNotUpdateRecord
203	ubReadOnly (Server Error)

UpdateHoldingRecord Function

The following describes the UpdateHoldingRecord function.

Required Parameters

The following are required parameters for the UpdateHoldingRecord function.

RecordID

Type: Long.

Values: A valid MARC holding record ID.

MARCRecord

Type: String.

Values: A valid MARC holding record.

DateTime

Type: VBDate.

Values: The date/time stamp the record was last updated. If there is no update date/time, then the create date/time is used (the update_date or create_date from the mfhd_master table).

CatLocationID

Type: Long.

Values: A valid Cataloging location ID.

RelatedRecordID

Type: Long.

Values: The linked bib ID (stored in the 004 field upon success).

HoldLocationID

Type: Long.

Values: A valid holding location ID such as the location ID that matches the location code in the 852 subfield b.

OpacSuppress

Type: Boolean.

Values: True / False.

Validation

The following is validated with the UpdateHoldingRecord function.

1. Validates the CatLocationID.
2. Validates the RelatedRecordID.
3. Validates that the record is a true MARC holding record.
4. Validates the data in 852 subfield b.
5. Validates that the record has a valid 004 field with a valid BibID.
6. Validates the RecordID.

Return Codes

The following return codes are provided for the UpdateHoldingRecord function.

UpdateHoldingReturnCode

Type: Integer.

Values: See Table 2-21.

Table 2-21. UpdateHoldingReturnCode Values

0	uhSuccess
1	uhUnknownError
2	uhNotAValidRecord
3	uhInvalidRecordType
7	uhRecordCanOnlyHaveOne001Field
8	uhRecordNeedsA001Field
9	uhRecordHasNo004Data
13	uhInvalidRecordID
15	uhInvalidLeaderLength
16	uhNoLeaderField
21	uhRecordEditError
23	uhInvalidLocationCode
24	uhInvalidCatalogingLocationID
30	uhRecordNeedsA004Field
31	uhCouldNotAssembleRecord
33	uhInvalid008Length
34	uhRecordHasNo008Data
35	uhLinkedIdError
36	uhRecordStructureIsInvalid
37	uhRecordHasNo852Field
38	uh852FieldHasNoSubfieldB
39	uh852SubfieldBHasNoData
41	uhDateTimeStampsDoNotMatch
42	uhCouldNotUpdateRecord

Table 2-21. UpdateHoldingReturnCode Values

203	uhReadOnly (Server Error)

UpdateItemData Function

The following describes the UpdateItemData function.

Required Parameters

The following are required parameters for the UpdateItemData function.

CatLocationID

Type: Long.

Values: A valid Cataloging location ID.

NOTE:
Other parameters need to be set in the cItem property of ClassBatchCat. See “Public Object of ClassBatchCat - cItem” on page 2-4 for more information.

Validation

The following is validated with the UpdateItemData function.

1. Validates the CatLocationID.
2. Validates the cItem.ItemID value.
3. Validates the cItem.PermLocationID value.
4. Validates the cItem.ItemTypeID value.
5. Validates that the cItem.TempTypeID Code is not zero, then validates that value.
6. Validates that the cItem.TempLocation is not zero, then validates that value.
7. Validates that the cItem.MediaTypeID is not zero, then validates that value.
8. Validates that the cItem.CopyNumber is between 0 and 32767 inclusive.
9. Validates that the cItem.PieceCount is between 0 and 9999 (inclusive).
10. Validates that the cItem.Price is numeric and not negative.

Return Codes

The following return codes are provided for the UpdateItemData function.

UpdateItemReturnCode

Type: Integer.

Values: See Table 2-22.

Table 2-22. UpdateItemReturnCode Values

0	uiSuccess
1	uiUnknownError
13	uiInvalidRecordID
24	uiInvalidCatalogingLocationID
42	uiCouldNotUpdateRecord
52	uiInvalidPrice
55	uiInvalidHoldingID
56	uiInvalidPermLocationID
57	uiInvalidTempLocationID
58	uiInvalidMediaTypeID
59	uiInvalidItemTypeID
60	uiInvalidTempTypeID

Index

A

C

D

E

F

G

H

I

M

P

R

S

T

U

Y