User's Guide

Integrated Library System

System Management: Cataloging

VIRTUA ILS – INTEGRATED LIBRARY SYSTEM

Virtua System Management

Cataloging User's Guide

Version 16.1

October 2017

1701 Kraft Drive
Blacksburg, Virginia 24060
U.S.A.
Phone 800.858.8857
E-mail: info@iii.com

Table of Contents

INDEX 200

1. Introduction

This user's guide documents the procedures for running server executables and scripts related to the Virtua Cataloging subsystem of the Virtua ILS – Integrated Library System. The information in this guide is intended for system administrators who feel comfortable working with programs that directly modify the contents of the Virtua database.

Most of the scripts and executables documented in this guide are not programs that you need to run on a daily basis. If you are not sure of whether you need to run a script or executable documented in this guide, contact an Innovative Customer Services representative. Additionally, if you do not feel comfortable running a particular script or executable, or if you are unaware of the consequences of running a script or executable, contact Innovative prior to running it.

Important:

- Since many of the scripts and executables documented in this guide modify data in the database, it is important that you have a recent export of your database.
- Most scripts in Virtua contain descriptive comments at the beginning of the file. Before you run a script, open it in a text editor and read ALL of the comments contained in the header.

Many of the scripts and executables discussed in this guide take a long time to run. For these programs, you may find it necessary to run them as background processes. For details, see the *Virtua System Management Reference Guide*.

This chapter covers the following topics:

1.1 Using this Guide

The *Virtua System Management: Cataloging User's Guide* contains instructions for performing system administration tasks related to the Cataloging subsystem of Virtua. You can use the list below and the table of contents to locate specific information in this guide.

For . . .	See . . .
Information about utilities that generate lists of record IDs from the database	Chapter 2
Instructions for using **write2709.exe**, Virtua's record extraction program, and other extraction utilities	Chapter 3
Instructions for generating formatted MARC records for optimized for viewing	Chapter 4
Instructions for using utilities that catalog, or process, records	Chapter 5
Instructions for working with authority records	Chapter 6
Instructions for modifying data in the bibliographic records in your database	Chapter 7
Information about scripts that deal with various aspects of item records	Chapter 8
Instructions for extracting a list of prediction patterns from the holdings record in your database	Chapter 9
Instructions for running conversion programs	Chapter 10
Instructions for enabling multiple subject headings	Chapter 11
Information about scripts for managing linked and FRBR records	Chapter 12
Information about scripts for modifying and identifying characters	Chapter 13
Information about scripts related to patrons	Chapter 14
Reference information about the scripts and executables documented in this guide	Appendix A
A list of changes specific to the Hong Kong Public Library	Appendix B

Reference table for mapping AACR2 elements to RDA in MARC records	Appendix C
A list of scripts and executables for use with Virtua's permalinks functionality	Appendix D
A list of changes that were made in the most recent update of this guide	Appendix E

1.2 Recommended Knowledge for Using this Guide

This user's guide is intended primarily for system administrators who are comfortable working with programs that directly modify the contents of the Virtua database. Throughout this guide, we assume that you have experience working with your server from the command line.

Additionally, throughout this guide, we assume that you have already read the *Virtua System Management Reference Guide*, which provides . . .

- Details on the directory of Virtua and the Virtua database.
- Administrative tips for working with UNIX, such as information on . . .
  - Running scripts and executables as background processes.
  - Redirecting output.
  - Detecting and killing processes.
  - Working with cron jobs.
- Administrative tips for working with Oracle, such as information on . . .
  - Accessing SQL*Plus.
  - Starting and shutting down the database.
  - Working with the Oracle Listener.
  - Managing passwords.
  - Exporting tables.
- Information on database backups.
- Details on character maps.
- Information on working with **psdriver.exe**.

Important: Information provided in the *Virtua System Management Reference Guide* is generally NOT repeated in this user's guide.

1.3 An Important Note about Environment Variables

Environment variables specify information about the working environment in the current UNIX session. Programs such as Virtua access and use environment variables when executing functions. Additionally, some of these variables are available for use from the command line.

Before you run any of the scripts or executables documented in this guide, you need to check the settings of at least three environment variables:

- **EXE_DIR** - Defines the path to the directory in which the scripts and executables for this version of Virtua are stored.
- **ORACLE_SID** - Defines the Oracle SID setting (i.e., *vtls01* or *vtls99*).
- **NLS_LANG** - Determines the character set used by Oracle.

If these variables are not set correctly, it is possible that you will run the wrong version of a program, modify the wrong database, or corrupt your data. Generally, these variables will be set for you in the **dbadmin** profile when you log in to the system, but we recommend that you double-check their setting before you run any scripts or executables. For information on setting and checking these and other environment variables such as VIRTUA_USER and VIRTUA_PASSWORD, see the *System Management Reference Guide*.

2. Extracting Record IDs

Several Virtua scripts and executables require that you provide a list of record IDs (001 tag) for each record on which you perform the requested action. For example, **write2709.exe** requires a record ID for each record that it extracts. This chapter provides instructions for running scripts that create lists of record IDs that you can use with other scripts and executables such as **write2709.exe** or **globalChange2.ksh**.

The following scripts are available for use in extracting record IDs based on the record type:

- **WriteAuthIdsFile.ksh** - Writes to a file the authority ID of each authority record in the database.
- **WritePermAuthIds.ksh** - Writes to a file the authority ID of each *permanent* authority record in the database.
- **WritePermSubjectIds.ksh** - Writes to a file the authority ID of each *permanent* subject authority record in the database.
- **PrintAuthoritybyState.ksh** - Writes to a file the authority ID of each authority record in the database with a given state.
- **WriteBibIdsFile.ksh** - Writes to a file the bibliographic ID of each bibliographic record in the database.
- **WriteHoldingIdsFile.ksh** - Writes to a file the holdings ID of each holdings record in the database.
- **WriteItemIdsFile.ksh** - Writes to a file the Item-ID of each item record in the database.

Other scripts are available for use in generating record IDs from other record IDs:

- **CreateBibIdsFromItemIds.sh**
- **CreateItemIdsFromBibIds.sh**
- **CreateItemIdsFromBarcodes.sh**
- **CreateItemBarcodesFromItemIds.sh**

Note: The additional script **WritePatronIdsFile.ksh** is documented in the *Virtua System Management/Circulation User's Guide*.

This chapter addresses the following topics:

2.1 Extracting Authority IDs

The script **WriteAuthIdsFile.ksh** writes to a file the authority IDs in the database.

To extract a list of authority IDs,

1. Log in to your server as **dbadmin**.
2. Type: **`WriteAuthIdsFile.ksh [output filename]`**

   Where **[output filename]** is the name of the file to which you want to write authority IDs.

   Note: If you specify the name of an existing file, Virtua will delete the contents of that file before writing authority IDs.
3. Press Enter.

   The script writes out the record IDs to the specified file.

2.2 Extracting Authority IDs of Permanent Authority Records

The script **WritePermAuthIds.ksh** writes to a file the authority IDs of every *permanent* authority record in the database.

To extract a list of authority IDs from permanent authority records,

1. Log in to your server as **dbadmin**.
2. Type: **`WritePermAuthIds.ksh [output filename]`**

   Where **[output filename]** is the name of the file to which you want to write authority IDs.

   Note: If you specify the name of an existing file, the software will delete the contents of that file before writing authority IDs.

3. Press Enter.

   The script writes out the record IDs to the specified file.

2.3 Extracting Authority IDs of Permanent Subject Authority Records

The script **WritePermSubjectIds.ksh** writes to a file the authority IDs of every *permanent subject* authority record in the database.

To extract a list of authority IDs from permanent subject authority records,

1. Log in to your server as **dbadmin**.
2. Type: **`WritePermSubjectIds.ksh [output filename]`**

   Where **[output filename]** is the name of the file to which you want to write authority IDs.

   Note: If you specify the name of an existing file, Virtua will delete the contents of that file before writing authority IDs.

3. Press Enter.

   The script writes out the record IDs to the specified file.

2.4 Extracting Authority IDs of Records with a Given State

The script **PrintAuthoritybyState.ksh** writes to a file the authority IDs of every record in the database with a given state. The resulting file will be sorted by bib level, tag, and heading.

To extract a list of authority IDs by state,

1. Log in to your server as **dbadmin**.
2. Type: **`PrintAuthoritybyState.ksh [state id]`**

   Note: If you use a log-in and password for your database, make sure the environment variables VIRTUA_USER and VIRTUA_PASSWORD are set before running the script
3. Press Enter

The script writes out the record IDs to the file **authorityRecords.print**.

2.5 Extracting Bibliographic IDs

2.5.1 Extracting All Bibliographic IDs

The script **WriteBibIdsFile.ksh** writes to a file the bibliographic IDs of every bibliographic record in the database.

To extract a list of bibliographic IDs,

1. Log in to your server as **dbadmin**.
2. Type: **`WriteBibIdsFile.ksh [output filename]`**

   Where **[output filename]** is the name of the file to which you want to write bibliographic IDs.

Note: If you specify the name of an existing file, Virtua will delete the contents of that file before writing bibliographic IDs.

3. Press Enter.

   The script writes out the record IDs to the specified file.

Important: The **WriteBibIdsFile.ksh** script should not be run concurrently with other scripts.

2.5.2 Extracting Masked Bibliographic IDs

The script **WriteMaskedBibIdsFile.ksh** writes to a file the bibliographic IDs of the masked bibliographic records in the database.

To extract a list of masked bibliographic IDs,

1. Log in to your server as **dbadmin**.

2. Type: **`WriteMaskedBibIdsFile.ksh [output filename]`**

   Where **[output filename]** is the name of the file to which you want to write masked bibliographic IDs.

   Note: If you specify the name of an existing file, Virtua will delete the contents of that file before writing the masked bibliographic IDs.

3. Press Enter.

   The script writes out the record IDs to the specified file.

Important: The **WriteMaskedBibIdsFile.ksh** script should not be run concurrently with other scripts.

2.6 Extracting Holdings IDs

The script **WriteHoldingIdsFile.ksh** writes to a file the holdings IDs of every holdings record in the database.

To extract a list of holdings IDs,

1. Log in to your server as **dbadmin**.
2. Type: **`WriteHoldingIdsFile.ksh [output filename]`**

   Where **[output filename]** is the name of the file to which you want to write holdings IDs.

   Note: If you specify the name of an existing file, the software will delete the contents of that file before writing holdings IDs.

3. Press Enter.

   The script writes out the record IDs to the specified file.

2.7 Extracting Item IDs

The script **WriteItemIdsFile.ksh** writes to a file the Item-IDs (001 tag) of every item record in the database.

To extract a list of Item-IDs,

1. Log in to your server as **dbadmin**.
2. Type: **`WriteItemIdsFile.ksh [output filename]`**

   Where **[output filename]** is the name of the file to which you want to write Item-IDs.

   Note: If you specify the name of an existing file, the software will delete the contents of that file before writing Item-IDs.

3. Press Enter.

   The script writes out the record IDs to the specified file.

2.8 Creating Bib IDs from Item IDs

The script **CreateBibIdsFromItemIds.sh** generates a file of Bib IDs from an input file of Item IDs.

Usage: `CreateBibIdsFromItemIds.sh inputFileItemIds`

The script produces one output file **BibIds.txt**, which will contain a distinct list of Bib IDs generated from the input file of Item IDs.

2.9 Creating Item IDs from Bib IDs

The script **CreateItemIdsFromBibIds.sh** generates a file of Item IDs from an input file of Bib IDs.

Usage: `CreateItemIdsFromBibIds.sh inputFileBibIds`

The script produces one output file **ItemIds.txt**, which will contain a distinct list of Item IDs generated from the mandatory input file of Bib IDs.

2.10 Creating Item IDs from Barcodes

The script **CreateItemIdsFromBarcodes.sh** generates a file of Item IDs from an input file of item barcodes.

Usage: `CreateItemIdsFromBarcodes.sh inputFileItemBarcodes`

The script produces one output file **ItemIds.txt**, which will contain a distinct list of Item IDs generated for all of the item barcodes in the input file.

2.11 Creating Barcodes from Item IDs

The script **CreateItemBarcodesFromItemIds.sh** generates a file of item barcodes from an input file of Item IDs.

Usage: `CreateItemBarcodesFromItemIds.sh inputFileItemIds`

The script produces one output file **ItemBarcodes.txt**, which will contain a distinct list of item barcodes generated from a mandatory input file of Item IDs.

3. Extracting Database Records

To extract records from the Virtua database in ISO-2709 format, use the executable **write2709.exe**. For a file of record IDs that you specify, **write2709.exe** program writes out the complete ISO-2709 MARC communications format record.

Note: For information on generating lists of record IDs that you can use with the **write2709.exe** program, see the chapter "Extracting Record IDs" in this user's guide.

To extract specific records or records for specific purposes from the Virtua database, a number of other methods are available.

This chapter covers the following topics:

3.1 Extracting Records Using write2709.exe

3.1.1 Running write2709.exe

The program **write2709.exe** extracts the entire ISO-2709 format MARC record for each record ID in the file that you specify. Generally, you will direct the output to a text file, which you can use later for input for a record loading program, such as **vload.exe**.

The basic format for running **write2709.exe** is:

write2709.exe [command options] < [file of IDs] > [output file]

Where . . .

- **[command options]** are the optional parameters that you can use to specify attributes such as record type or date range. For a description of the available

command-line options, see the section "Command-line Options for write2709.exe" in this chapter.

- **[file of IDs]** is the name of the file which lists, line-by-line, the record IDs of the records that you want to extract.
- **[output file]** is the file to which you want **write2709.exe** to extract records.

Note: Records in Error state are NOT extracted by **write2709.exe**.

To get online help regarding options to be used with **write2709.exe**, type:

```
write2709.exe -help
-OR-
write 2709.exe -?
```

3.1.2 Command-line Options for write2709.exe

This section discusses each of the options that are available for use with **write2709.exe**. You can specify these options on the command line as described in the previous section. Before running this program, we recommend that you read the contents of this section.

For each command-line option, the following information is provided:

- **Description** - A brief explanation of what the option does (i.e., how it affects the extraction).
- **Format** - The format that you must use to call the option at the command line.
- **Default Value** - The default value that Virtua uses if the option is NOT specified.

  Note: If you want to use the default value, you do not need to call the option when running **write2709.exe**.
- **Example** - An example of the syntax that you will use to call the option on the command line or in a load options file.

3.1.2.1 Specifying the Type of Records to Extract (-T)

Description: Specifies the type of records that you want **write2709.exe** to extract.

Format: `-T[ID code]`

Where **[ID code]** is one of the following:

- **101** - Bibliographic records
- **102** - Authority records
- **103** - Heading records **Tip:** Generally, this option will be used only when **write2709.exe** is called by another script.
- **104** - Holdings records
- **105** - Patron records.

Default: 101 (Bibliographic)

Example: If you want to extract patron records, type:

```
write2709.exe -T105 < input.in > output.out
```

The program will extract the patron records specified in the file **input.in** and write the entire contents of each record to a file named **output.out**.

3.1.2.2 Extracting Records Created or Last Modified by a Specific User (-O)

Description: Lets you extract all records that were created or *last* modified by a given user.

Format: **`-O[Username]`**

Where **[Username]** is the username for a Virtua user profile.

Default: None. If you do not specify this option, Virtua extracts all records regardless of user.

Example: If you want to extract all records created or last modified by the *staff* user, type:

```
write2709.exe -Ostaff < input.in > output.out
```

3.1.2.3 Specifying an Output Character Set Encoding (-C)

Description: Lets you choose a character set encoding for the output.

Format: **`-C[character set code]`**

Where **[character set code]** is one of the following codes:

- **2** - UTF-8
- **10** - MARC-8 (MARC 21)
- **11** - ANSI Z39.47 (ANSI-8)
- **12** - EUROPA-3
- **13** - Windows Latin1
- **14** - PC-8
- **15** - ALA
- **16** - Windows Arabic
- **17** - Windows Hebrew
- **18** - Windows Cyrillic
- **19** - Windows Latin2
- **20** - ISO 6937-2
- **21** - CP-850 (Microsoft Code Page)
- **22** - ISO 6937-2 + Arabic
- **23** - ISO 6937-2 + Greek
- **24** - Big5 (Tamkang version)
- **25** - ANSI-8 + Hebrew
- **26** - UTF-8 character set where some separate diacritics need to be combined
- **28** - ANSI Z39.47 (ANSI-8) - Swiss version
- **30** - GBK (encoding of CJK characters)
- **31** - TIS620 Classic Thai
- **32** - ISO 5426 (International Serials Data System interchange)
- **33** - CCCII (Chinese Character Code for Information Interchange)
- **34** - GB 18030 (Chinese National Standard GB 18030-2000)
- **35** - Big5-HKSCS (Hong Kong Special Character Set)

Default: 2 (UTF-8)

Example: If you want to extract records using ISO 6937-2 encoding, type:

```
write2709.exe -C20 < input.in > output.out
```

The program will use ISO 6937-2 encoding to extract the records specified in the file **input.in** and write the entire contents of each record to a file named **output.out**.

If you do not use the **-C** option in the command, **write2709.exe** uses UTF-8 encoding.

3.1.2.4 Extracting Records by Begin Date (-B/-Y)

You can extract records with a date *later* than a specified date by using two options:

- **-B** - Specifies the begin date.
- **-Y** - Specifies the tag and subfields on which to match the begin date.

When you specify these options on the command line, **write2709.exe** will extract from the database all records that contain in the specified tag (**-Y**) a date that is *later* than the specified begin date (**-B**). We describe these options in detail in the following sections.

Hint: You can use these options in combination with the end date options to extract records within a particular date *range*. For details, see the section "Extracting Records by End Date (-E/-Z)" in this chapter. Additionally, see the section "Specifying the Latest Date Option (-L)" for information on another option that you can use when specifying both the **-B** and **-E** options.

3.1.2.4.1 Specifying a Begin Date (-B)

Description: Lets you extract records with a date *later* than the date specified.

Note: When you use the **-B** option you must also use the **-Y** option to tell **write2709.exe** the tag on which to match the specified date. For details, see the section "Defining Which Tag to Use with the Begin Date Option (-Y)" in this chapter.

Format: `-B[year]:[month]:[day]:[hour]:[minute]`

Where **[year]** is the four-digit year, **[month]** the two-digit month, **[day]** the two-digit day, **[hour]** the two-digit hour, and **[minute]** the two-digit minute.

Default: None

Example: To extract all records with a date *later* than January 1, 2004 in the 039 tag, subfield $y (no offset), use this option:

```
-B2004:01:31:00:00 -Y039:y:0
```

3.1.2.4.2 Defining Which Tag to Use with the Begin Date Option (-Y)

Description: Defines the tag and subfield that will be used with the **-B** option.

Format: **`-Y[tag]:[subfield]:[offset]`**

Where . . .

- **[tag]** is a three-digit MARC tag.
- **[subfield]** is a subfield in the tag you specified.
- **[offset]** is the number of positions from the beginning of the tag that the date begins.

Example: To extract all records with a date *later* than January 1, 2004 in the 039 tag, subfield $y (no offset), use this option:

```
-B2004:01:31:00:00 -Y039:y:0
```

Note: Virtua uses the first occurrence of the tag/subfield you specify.

3.1.2.5 Extracting Records by End Date (-E/-Z)

You can extract records with a date *earlier* than a specified date by using two options:

- **-E** - Specifies the end date.
- **-Z** - Specifies the tag and subfields on which to match the end date.

When you specify these options on the command line, **write2709.exe** will extract from the database all records that contain in the specified tag (**-Z**) a date that is *earlier* than the specified end date (**-E**). We describe these options in detail in the following sections.

Hint: You can use these options in combination with the begin date options to extract records within a particular date *range*. For details, see the section "Extracting Records by Begin Date (-B/-Y)" in this chapter. Additionally, see the section "Specifying the Latest Date Option (-L)" for information on another option that you can use when specifying both the **-B** and **-E** options.

3.1.2.5.1 Specifying an End Date (-E)

Description: Lets you extract records with a date *earlier* than the date specified.

Note: When you use the **-E** option you must also use the **-Z** option to tell **write2709.exe** the tag on which to match the specified date. For details, see the section "Defining Which Tag to Use with the End Date Option (-Z)" in this chapter.

Format: **`-E[year]:[month]:[day]:[hour]:[minute]`**

Where **[year]** is the four-digit year, **[month]** the two-digit month, **[day]** the two-digit day, **[hour]** the two-digit hour, and **[minute]** the two-digit minute.

Default: None

Example: To extract all records with a date *earlier* than January 31, 2004 in the 039 tag, subfield $y (no offset), use this option:

```
-E2004:01:31:00:00 -Z039:y:0
```

3.1.2.5.2 Defining Which Tag to Use with the End Date Option (-Z)

Description: Defines the tag and subfield that will be used with the **-E** option.

Format: **`-Z[tag]:[subfield]:[offset]`**

Where . . .

- **[tag]** is a three-digit MARC tag.
- **[subfield]** is a subfield in the tag you specified.
- **[offset]** is the number of positions from the beginning of the tag that the date begins.

Example: To extract all records with a date *earlier* than January 31, 2004 in the 039 tag, subfield $y (no offset), use this option:

```
-E2004:01:31:00:00 -Z039:y:0
```

Note: Virtua uses the first occurrence of the tag/subfield you specify.

3.1.2.6 Specifying the Latest Date Option (-L)

Description: When two different dates are used to extract records for a given date range, lets you extract the record with the latest date. When you use this option, **write2709.exe** gets from each record the latest of the dates in the tags defined by the **-Y** and **-Z** options (see the sections "Defining Which Tag to Use with the Begin Date Option (-Y) and "Defining Which Tag to Use with the End Date Option (-Z)"). If this date is between the dates you defined with the **-B** and **-E** options (see the sections "Specifying a Begin Date (-B)" and "Specifying an End Date (-E)"), the record will be extracted.

A typical use of this function is to extract a record only if it has been last modified within a specified date range. But records that have never been modified have only a create date (039 subfield $y) and not a last modified date (039 subfield $a). The **-L** option allows **write2709.exe** to take the most recent of the two dates. If one date does not exist, the remaining date is used.

Note: To use this option, you must also use the **-B** and **-E** options.

Format: -L

Default: None

Examples:

- To extract all records that have a date in the 039 tag subfield $a that is between January 1, 2001 and December 31, 2001, type:

```
write2709.exe -
B2001:01:01:00:00 -E2001:12:31:00:00 -Y039:a:0 -Z039:a:0 <
input.in > output.out
```

- To get the later of the two dates in the 039 tag subfield $y and 039 tag subfield $a and extract records where that date is between August 16, 1999 and July 19 2002, type:

```
write2709.exe -L -
B1999:08:16:00:00 -E2002:07:19:2002:00:00 -Y039:y:0 -Z039:a:0 <
input.in > output.out
```

In this example, **write2709.exe** looks at the dates in the 039 tag subfield $a (last modified date) and in the 039 tag subfield $y (created date) to obtain the latest of the two. If the 039 tag subfield $a does not exist, meaning that the record has not been modified since creation, the program will use the date in the 039 tag subfield $y.

3.1.2.7 Extracting Only Record IDs (-F)

Description: Lets you extract only the record ID of each record.

Format: `-F`

Default: None. By default, this option is not used.

Example: To extract the record ID of bibliographic records created after January 18, 2002, type:

```
write2709.exe -F -B2002:01:18:00:00 -Y039:y:0 < input.in > output.out
```

For each record ID listed in the file **input.in** that was created after the given date, the program will write the record ID to a file named **output.out**.

3.1.2.8 Deleting Tags from the Extracted Records (-D)

Description: Lets you specify that one or more tags should be deleted from the records that are extracted.

Format: `-D[tag definition filename]`

Where **[tag definition filename]** is the name of the file that contains information about the tags to delete.

The file that defines the tags to be deleted from extracted records contains a line for each tag definition. The tag definition is in the following format:

Where . . .

- **[Tag]** is the three-digit tag number.

- **[Indicator 1]** is the one-character first indicator.
  -OR-
  An * if you want to specify all indicators.
  -OR-
  A blank space if you want to specify no indicators.

1. **[Indicator 2]** is the one-character second indicator.
   -OR-
   An * if you want to specify all indicators.
   -OR-
   A blank space if you want to specify no indicators.

> **Hint:** Notice that there is a blank space between the tag and the first indicator.

Default: None. By default, no tags are deleted from the extracted records.

Example: To delete the 025 tag (no indicators), the 039 tag (second indicator of 9), and the 590 tag (first indicator of 2 and second indicator of 1), from each record you extract, type:

```
025
039  9
590 21
```

In the second line of the example, there is an extra space before the second indicator of 9 that represents the blank first indicator.

3.1.2.9 Replacing Strings in Authority Records (-X, -S, -R)

Description: Lets you specify that in each extracted authority record, a string in a given tag should be replaced with another string. The following command line options are used to replace text in an authority record:

Format: `-X[n] -S["old string"] -R["new string"]`

Where . . .

- **-X[n]** - Defines the tag range in which the text replacement will take place, where **n** is one of the following values:
  - **1** - 1xx
  - **4** - 4xx
  - **5** - 5xx

  Note: If you do not use the **-X** option in the command, the replace will take place in the default tag range of 1xx.
- **-S["old string"]** - Specifies the text that will be replaced, where **"old string"** is a character string that consists of no more than 60 characters. If a record does not have the specified string, it will NOT be extracted.
- **-R["new string"]**- Specifies the new text that will replace the text defined in the **-S** option. This parameter accepts quoted text that consists of no more than 60 characters.

Note: In addition to the above options, you must include the **T102** option to indicate that the records processed are authority records.

Default: None

Example: To extract records with the text *Bill Gates* in the 5xx tag and replace it with the text *Bill Clinton*, type:

```
write2709.exe -T102 -X5 -S"Bill Gates" -R"Bill Clinton"
< input.in > output.out
```

3.1.2.10 Writing Item Information to the 949 Tag (-A)

Description: Specifies that **write2709.exe** should write information about each attached item in the 949 tag of each bibliographic record. If you use this option, extracted bibliographic records will have a 949 tag for each attached item. Each 949 tag can contain the following subfields, depending on the available information:

- **$6** - Item barcode
- **$9** - Unit information
- **$a** - First item-level call number
- **$b** - Second item-level call number
- **$e** - Accession number
- **$d** - EPN
- **$o** - Staff note
- **$p** - Public note
- **$q** - Check-in note

- **$r** - Check-out note
- **$s** - Item status
- **$D** - Location code
- **$F** - Copy number
- **$G** - Temporary shelf location
- **$X** - Item class

Note: If you use the **-A** option, **write2709.exe** will first delete the 949 tag from ALL extracted records before adding new 949 tags. Existing 949 tags will be deleted even if the record does not have attached items.

Format: `-A`

Default: None. By default, the **-A** option is not used, and item information is not extracted.

Example:

```
write2709.exe < input.in > output.out -T101 -A
```

3.1.2.11 Writing Item Information to the 852 Tag (-I)

Description: Specifies that **write2709.exe** should write information about each attached item in the 852 tag of each bibliographic record. If you choose this option (**-I**), extracted bibliographic records will have an 852 tag for each attached item. Each 852 tag will contain the following subfields (assuming the related information exists in the item record):

- **$b** - Location code
- **$h** - First item-level call number
- **$t** - Copy number
- **$z** - Item barcode

Note:
- If you use the **-I** option, **write2709.exe** will first delete the 852 tag from ALL extracted records before adding new 852 tags. Existing 852 tags will be deleted even if the record does not have attached items.
- This program does not write out the second item-level call number. If no first item-level call number exists for an item, the tag will not include call number information.

Default: By default, item information is NOT extracted to the 852 tag.

Example:

```
write2709.exe < input.in > output.out -T101 -I
```

3.1.2.12 Writing Item Information to the 852 Tag: Option (-N)

Description: Specifies that **write2709.exe** should write information about each attached item in the 852 tag of each bibliographic record. If you choose this option (**-N**), extracted bibliographic records will have an 852 tag for each attached item. Each 852 tag will contain the following subfields (assuming the related information exists in the item record):

- **$a** - MARC organization code
- **$b** - Location name
- **$h** - Main call number (information up to the first space in the first item-level call number)
- **$i** - Call number cutter (all information after the first space in the first item-level call number)
- **$k** - Call number prefix
- **$m** - Call number suffix
- **$p** - Item barcode

Note:
- If you use the **-N** option, **write2709.exe** will first delete the 852 tag from ALL extracted records before adding new 852 tags. Existing 852 tags will be deleted even if the record does not have attached items.
- This program does not write out the second item-level call number. If no first item-level call number exists for an item, the tag will not include call number information.

Default: By default, item information is NOT extracted to the 852 tag.

Example:

```
write2709.exe < input.in > output.out -T101 -N
```

3.1.2.13 Writing Holdings Item Information to the 852 Tag (-G)

Description: Specifies that **write2709.exe** should write information about each item attached to a holdings record in the 852 tag of the parent bibliographic record. If you choose this option (**-G**), extracted bibliographic records will have an 852 tag for each item attached to a holding. There will be one 852 tag for each item attached to each holding attached to the bibliographic record. Each 852 tag will contain the following subfields (assuming the information exists in the holdings/item record):

- **$a** - MARC organization code (from the 852 tag subfield $a of each holdings record).
- **$b** - Location name.
- **$h** - Main call number (information up to the first space in the first item-level call number, or if no item-level call number exists, the 852 tag subfield $h from the holdings record).
- **$i** - Call number cutter (all information after the first space in the first item-level call number, or if no item-level call number exists, the 852 tag subfield $h from the holdings record).
- **$m** - Call number suffix (all information after the second space in the first item-level call number, or if no item-level call number exists, the 852 tag subfield $m from the holdings record).
- **$p** - Item barcode.
- **$x** - Public notes from the 9xx tags of the copy-specific note record attached to the item or holding.
- **$z** - Private notes from the 9xx tags of the copy-specific note record attached to the item or holding.

Notes:

- If no items are attached to the holding record, one 852 tag will be added without the subfield $p.
- If a location filter list is specified via the **-l** option, the **–G** option will write information to the 852 tag for the items attached to holdings from *only* those locations specified in the list. (See the section "Using a Location Filter List (-l)" in this user's guide for details.)

Default: By default, holdings/item information is NOT extracted to the 852 tag.

Example:

```
write2709.exe < input.in > output.out -T101 -G
```

3.1.2.14 Writing Holdings Information to the 852 Tag (-J)

Description: Specifies that **write2709.exe** should write information about each holding attached to a bibliographic record in the 852 tag of that bibliographic record. If you choose this option (**-J**), extracted bibliographic records will have an 852 tag for each holding attached to the record. There will be one 852 tag for each holding attached to the bibliographic record. Each 852 tag will contain the following subfields (assuming the information exists in the holdings record):

- **$a** - MARC organization code (from the 852 tag subfield $a of each holdings record).
- **$b** - Location name.

Note: If a location filter list is specified with the **-l** option, the **–J** option will write information to the 852 tag for the holdings attached to records from *only* those locations specified in the list. (See the section "Using a Location Filter List (-l)" in this user's guide for details.)

Default: By default, holdings information is NOT extracted to the 852 tag.

Example:

```
write2709.exe < input.in > output.out -T101 -J
```

3.1.2.15 Extracting Holdings Records Following Each Bibliographic Record (-K)

Description: Lets you specify, for records where the ID type is bibliographic, that the bibliographic record should be output followed immediately by each attached holdings record.

Note: If a location filter list is specified with the **-l** option, the **-K** option will extract the holdings only from those locations specified in the list. (See the section "Using a Location Filter List (-l)" in this user's guide for details.)

Default: By default, this option is not used

Example:

```
write2709.exe < input.in > output.out -T101 -K
```

3.1.2.16 Using a Location Filter List (-l)

Description: Lets you specify a file of location IDs to use along with the following options: **-G, -J,** and **-K**. The actions indicated in these options will be applied to records *only* at the locations specified by the IDs listed in the file.

Format: -l[filename]

Where the **[filename]** is the name of a file containing a list of location ID numbers separated by any amount of white space (e.g., one location ID per line).

Default: None. By default, this option is not used.

Example:

```
write2709.exe < input.in > output.out -T101 -G -l[filename]
```

3.1.2.17 Extracting Records to Be Imported into a VTLS Classic Database (-V)

Description: Lets you extract records in a format that can be loaded into a VTLS Classic database.

Format: -V

Default: None. By default, this option is not used.

Example:

```
write2709.exe < input.in > output.out -T101 -V
```

Note: When you extract bibliographic or authority records using the **-**V option, the records will contain a 999 VTLSFF tag that holds the local level, entry date, and bib level.

3.1.2.18 Extracting Records in ISSN Format (-M)

Description: Lets you extract records in ISSN format.

Format: `-M[ISSN Center]`

Where **[ISSN Center]** is a two-character ISSN Center code that will be inserted in positions 18 and 19 of the 008 tag of each extracted record.

Default: None. By default, this option is not used.

Example:

```
write2709.exe < input.in > output.out -T101 -M34
```

Note: If you use this option, the program will extract records using ISDS character mapping. Do not use the **-C** option in your command.

3.1.2.19 Setting the Bibliographic Level to s (-W)

Description: Lets you set the bibliographic level to **s** (serial) in all extracted records. If -W is specified and the MARC 21 (ISSN Version) flag is set, bibliographic records will be converted to "pure" MARC 21 format (i.e., the normal MARC 21 format as opposed to the internal MARC version under the ISSN flag) before extraction.

Format: `-W`

Default: None. By default, this option is not used.

Example:

```
write2709.exe < input.in > output.out -T101 -W
```

3.1.2.20 Extracting the Base Record If an Error Record Exists (-b)

Description: Tells the system always to extract the base record if any special state (e.g. Error state) record exists.

Format: `-b`

Defaults:

- **For bibliographic records**: Do not extract the base record if an Error state record exists. Instead, extract the base record if any other special state record exists.
- **For authority records**: Do not extract the base record if any special state record exists.
- **For patron records**: Always extract the base record regardless of the flag.
- **For holdings records**: Extract special state record if it exists.

Example:

```
write2709.exe < input.in > output.out -T101 -b
```

3.1.2.21 Extracting Only XML Records (-x)

Description: Lets you extract existing bibliographic XML records into a new XML file. This option can also be used as **[--extract-marc-records-only]**

Format: -x

Default : None. By default, this option is not used.

Example:

```
write2709.exe -x
```

3.1.2.22 Translating Codes for the European Union Publication Office (-c)

The European Union Publications Office and its external contractors use codes, which are entered into the MARC bibliographic record, to represent a word, sentence or agency. For example, AMOC means Agency for the Management of Operational Cooperation. When these records are extracted, the codes will require translation to values specified by the language stored in tag 041 subfield $a of the record.

Note: write2709.exe supports this option and outputs it in the usage statement only if the database license key supports codes.

Description: Tells the system to export the bibliographic record with the code translated into the language specified in the record. If there is no language specified in the record, the default language will be used.

Format: `-c`

Default: None. By default, this option is not used.

Example:

```
write2709.exe < input.in > output.out -T101 –c
```

3.1.2.23 Translating Codes to a Given Language for the European Union Publication Office (-d)

The European Union Publications Office and its external contractors use codes, which are entered into the MARC bibliographic record, to represent a word, sentence or agency. For example, AMOC means Agency for the Management of Operational Cooperation. When these records are extracted, the codes will require translation to values specified by the language stored in tag 041 subfield $a of the record *unless* you define a specific language in the argument used with option **–d [--language_code]**.

Note: write2709.exe supports this option and outputs it in the usage statement only if the database license key supports codes.

Description: Tells the system to export the bibliographic record with the code translated into the language specified in the argument for option -d. *Option -d must be used with option -c.*

Format: `-d[language code]`

Where **[language code]** can consist of either two or three characters, depending on the language code used in the database. If there is not a translation in the specified language, the default translation is used, and if this is not present, the second default translation is used.

Default: None. By default, this option is not used.

Example:

```
write2709.exe < input.in > output.out -T101 –c –deng
```

3.1.2.24 Translating Only Specific Codes to a Given Language for the European Union Publication Office (-e)

Note: write2709.exe supports this option and outputs it in the usage statement only if the database license key supports codes.

Description: The **--codes_translate_add_901_tag** option tells the system to translate code only from tags 440 $a and 773 $t into the language of tag 041 $a. If there is no translation in the tables or the record has more than one 041 $a or the value of the 041 $a is MUL (for multilingual), then the record is not decoded AND a new 901 tag is created with the text *Non-decoded 440/773* in the subfield $a.

Format: `-e`

Default: None. By default, this option is not used.

Example:

```
write2709.exe < input.in > output.out -T101 –e
```

3.1.3 Command Line Reference Table for write2709.exe

The following table details the command-line options for **write2709.exe**. Before you use these options, you should read the detailed instructions for each option in the section "Command-line Options for write2709.exe" in this user's guide.

Option Code	Description	Values (default value is italicized)	Notes
-T	Determines the type of records that are being extracted.	***101*** - *Bibliographic records* **102** - Authority records **103** - Heading records **104** - Holdings records **105** - Patron records.	
-O	Defines the username that must appear in the 039 $z or 039 $b.	Any username from a Virtua user profile.	

Option Code	Description	Values (default value is italicized)	Notes
-C	Determines the character set encoding that the program will use to write the extracted records.	***2*** *- UTF-8* **10** - MARC-8 **11** - ANSI Z39.47 **12** - EUROPA-3 **13** - Windows Latin1 **14** - PC-8 **15** - ALA **16** - Windows Arabic **17** - Windows Hebrew **18** - Windows Cyrillic **19** - Windows Latin2 **20** - ISO6937-2 **21** - Microsoft CP-850 **22** - ISO 6937-2 + Arabic **23** - Greek **24** - Big5 **25** - ANSI-8 + Hebrew **26** - UTF-8 character set special **28** - ANSI-8 - Swiss version **30** - GBK **31** - TIS620 Classic Thai **32** - ISO 5426 (International Serials Data System interchange) **33** - CCCII **34** - GB 18030 **35** - Big5-HKSCS	
-B	Determines the starting date of the date range.	Any date using the following format: **YYYY:MM:DD:HH:MN**	Read the section "Specifying a Begin Date" before using this option.
-E	Determines the ending date of the date range.	Any date using the following format: **YYYY:MM:DD:HH:MN**	Read the section "Specifying an End Date" before using this option.
-Y	Defines the tag to which the **-B** option refers.	Any tag that specifies a date. *(Default - 039 subfield $a)*	Read the section "Extracting Records by Begin Date" before using this option.

Option Code	Description	Values (default value is italicized)	Notes
-Z	Defines the tag to which the **-E** option refers.	Any tag that specifies a date. *(Default - 039 subfield $a)*	Read the section "Extracting Records by End Date" before using this option.
-L	Instructs the program to use the latest of the dates in the tags defined for the **-Y** and **-Z** options when checking the date range.	N/A	Read the section "Specifying the Latest Date Option" before using this option.
-F	Outputs just the record ID of each extracted record.	N/A	
-D	Deletes specific tags from the extracted records.	A filename containing tag definitions.	
-X	Determines the tag group in which the program will look for the text defined in the **-S** option.	***1*** *- 1xx* **4** - 4xx 5xx	Read the section "Replacing Strings in Authority Records" before using this option.
-S	Specifies the text that will be replaced by the text defined by the **-R** option.	Any quoted text that is no more than 60 characters.	Read the section "Replacing Strings in Authority Records" before using this option.
-R	Specifies the text that will replace the text defined by the **-S** option.	Any quoted text that is no more than 60 characters.	Read the section "Replacing Strings in Authority Records" before using this option.

Option Code	Description	Values (default value is italicized)	Notes
-A	Instructs the program to write out item information to the 949 tag.	N/A	The following subfields are included in the generated 949 tag: $6 - Item barcode $9 - Unit information $a - First item-level call number $b - Second item-level call number $d - EPN $e - Accession number $o - Staff note $p - Public note $q - Check-in note $r - Check-out note $s - Item status $D - Location code $F - Copy number $X - Item class
-I	Instructs the program to write out item information to the 852 tag.	N/A	The following subfields are included in the generated 852 tag: $b - Location code First item-level call number $t - Copy number $z - Item barcode
-N	Instructs the program to write out item information to the 852 tag.	N/A	The following subfields are included in the generated 852 tag: $a - MARC organization code $b - Location name $h - Main call number $i - Call number cutter $k - Call number prefix $m - Call number suffix $p - Item barcode
-V	Formats extracted records so that they can be imported into a VTLS Classic database.	N/A	

Option Code	Description	Values (default value is italicized)	Notes
-M	Converts extracted records into ISSN format.	A two-character code. This code is inserted in positions 18 and 19 of the 008 tag.	Customization for Swiss National Library.
-W	Sets the bibliographic level to **s** in all extracted records.	N/A	
-b	Determines if the base state record is extracted when any special state (e.g. Error state) record exists.	For the list, see "Extracting the Base Record If an Error Record Exists (-b)"	
-c	Translates codes in extracted bibliographic records per language specified in the record.	N/A	Customization for European Union Publications Office.
-d	Translates codes in extracted bibliographic records per language specified in the argument at the command line.	N/A	Customization for European Union Publications Office.
-e	Only translates code from tags 440 $a and 773 $t into the language of tag 041 $a. If there is no translation in the tables or the record has more than one 041 $a or the value of the 041 $a is MUL (for multilingual), then the record is not decoded AND a new 901 tag is created with the text *Non-decoded 440/773* in the subfield $a.	N/A	For EUPO users only

3.2 Extracting Records Using Other Utilities

This section discusses other utilities available for extracting records from the database.

3.2.1 Extracting "Deleted State" Bibliographic Records to a File

You can extract to a file MARC 21 (UTF-8) bibliographic records in a Virtua database that are in Deleted state by running the perl script **extract_deleted_bibs.pl**.

The following options are available with the script:

- Extract all deleted records stored in the state_record_bibliographic table (*default*).
- Extract deleted records based on a specified Start and End date.
- Change the leader 05 value of all records to 'd' before outputting the records.

The script accepts the `VIRTUA_USER, VIRTUA_PASSWORD, ORACLE_HOME, and LIBPATH` environment settings for configuration. If they are not set, the following defaults are set by the script:

```
VIRTUA_USER = 'syscli'
VIRTUA_PASSWORD = 'syscli'
ORACLE_HOME = '/usr/vtls/virtua/ora11'
LIBPATH = '$ORACLE_HOME/lib:/lib:/usr/lib'
```

To run extract_deleted_bibs.pl,

1. Log in to your server as **dbadmin.**
2. At the prompt, type: **`extract_deleted_bibs.pl --help`**

A list of usage parameters that the script accepts appears:

```
Usage: extract_deleted_bibs.pl [--start-date "YYYY-MM-DD HH:MI:SS"]
[--end-date "YYYY-MM-DD HH:MI:SS"] [--no-status-change] [--help]

Example 1: extract_deleted_bibs.pl (extracts all deleted bibs and
changes status to 'd')

Example 2: extract_deleted_bibs.pl --no-status-change
```

```
Example 3: extract_deleted_bibs.pl --start-date 2005-01-30

Example 4: extract_deleted_bibs.pl --start-date 2005-01-30 --end-
date 2005-01-31

Example 5: extract_deleted_bibs.pl --start-date "2005-01-30
01:00:00" --end-date "2005-01-31 23:59:59"

Example 6: extract_deleted_bibs.pl --xml-records >
extractedxmlbibs.txt
```

3. At the prompt, type: **extract_deleted_bibs.pl**
   and include any applicable usage parameters in the command line.

   The software extracts to a file the delete state records according to the specifications in your usage parameters. Note that the software does NOT delete the records from the database.

3.2.2 Extracting Bibliographic Records Based on Creation Date

You can extract MARC 21 (UTF-8) bibliographic records in a Virtua database based on creation date by running the script **extract_bibs_by_creation_date.sh**. The script first creates a file of bibliographic IDs of records created after a specified date, and then the script runs **write2709.exe** to extract the records.

To run extract_bibs_by_creation_date.sh,

1. Log in to your server as **dbadmin**.
2. At the prompt, type: **extract_bibs_by_creation_date.sh**
3. Respond to the interactive prompts for the date.
4. Press Enter.

Virtua creates three files:

- **BibId_{date}.ids** - A file of Bib-IDs
- **Bibs_{date}.rec** - A file of bibliographic records
- **write2709_{date}.err** - A file of errors produced by the program

Where **{date}** = the timestamp when the file was created.

3.2.3 Extracting Error State Bibliographic Records

You can extract MARC bibliographic records in a Virtua database that are in Error state by running the perl script **extract_bib_in_error_state.pl**.

The following options are available with the script:

- Extract all Error state records stored in the state_record_bibliographic table (*default*).
- Extract Error state records based on a specified Start and End date.

The script accepts the `VIRTUA_USER, VIRTUA_PASSWORD, ORACLE_HOME, and LIBPATH` environment settings for configuration. If they are not set, the following defaults are set by the script:

```
VIRTUA_USER = 'syscli'
VIRTUA_PASSWORD = 'syscli'
ORACLE_HOME = '/usr/vtls/virtua/ora11'
LIBPATH = '$ORACLE_HOME/lib:/lib:/usr/lib'
```

To run extract_bibs_in_error_state.pl,

1. Log in to your server as **dbadmin.**
2. At the prompt, type: **`extract_bibs_in_error_state.pl --help`**

A list of usage parameters that the script accepts appears:

```
Usage: extract_bibs_in_error_state.pl [--start-date "YYYY-MM-DD
HH24:MI:SS"] [--end-date "YYYY-MM-DD HH24:MI:SS"] [--help]
>outputfile

Example 1: extract_bibs_in_error_state.pl (extracts all
bibliographic records in Error state)

Example 2: extract_bibs_in_error_state.pl --start-date 2014-01-30

Example 3: extract_bibs_in_error_state.pl --start-date 2014-01-30 -
-end-date 2014-02-25

Example 4: extract_bibs_in_error_state.pl --start-date 2014-01-30
01:00:00 --end-date 2014-02-25 23:59:59
```

4. At the prompt, type: **`extract_bibs_in_error_state.pl`**
   and include any applicable usage parameters in the command line.

The software extracts the error state records as specified in your usage parameters and prints the output to the screen. If you want the data stored in a file, redirect the output into a file of your choosing. For example, to extract all bib records in Error state into file **bib.mrc**, type:

```
extract_patrons_in_error_state.pl >bib.mrc
```

3.2.4 Extracting Error State Patron Records

You can extract and/or delete MARC 21 (UTF-8) patron records in a Virtua database that are in Error state by running the perl script **extract_patrons_in_error_state.pl**.

The following options are available with the script:

- Extract all error state records stored in the state_record_patron table (*default*).
- Extract error state records based on a specified Start and End date.
- Extract and/or delete all error state records stored in the state_record_patron table.
- Extract and/or delete all error state records stored in the state_record_patron table based on a specified Start and End date.

The script accepts the `VIRTUA_USER, VIRTUA_PASSWORD, ORACLE_HOME, and LIBPATH` environment settings for configuration. If they are not set, the following defaults are set by the script:

```
VIRTUA_USER = 'syscli'
VIRTUA_PASSWORD = 'syscli'
ORACLE_HOME = '/usr/vtls/virtua/ora11'
LIBPATH = '$ORACLE_HOME/lib:/lib:/usr/lib'
```

To run extract_patrons_in_error_state.pl,

3. Log in to your server as **dbadmin.**
4. At the prompt, type: **`extract_patrons_in_error_state.pl --help`**

A list of usage parameters that the script accepts appears:

```
Usage: extract_patrons_in_error_state.pl [--start-date \"YYYY-MM-DD
HH:MI:SS\"] [--end-date \"YYYY-MM-DD HH:MI:SS\"] [--delete-only] [-
-delete-and-extract] [--help]

Example 1: extract_patrons_in_error_state.pl (extracts all patrons
in error state)
```

```
Example 2: extract_patrons_in_error_state.pl --start-date 2005-01-
30

Example 3: extract_patrons_in_error_state.pl --start-date 2005-01-
30 --end-date 2005-01-31

Example 4: extract_patrons_in_error_state.pl --start-date 2005-01-
30 01:00:00 --end-date 2005-01-31 23:59:59

Example 5: extract_patrons_in_error_state.pl --extract-and-delete
(extracts and deletes all patrons in error state)

Example 6: extract_patrons_in_error_state.pl --start-date 2005-01-
30 --delete-only

Example 7: extract_patrons_in_error_state.pl --start-date 2005-01-
30 --end-date 2005-01-31 --delete-and-extract
```

5. At the prompt, type: **extract_patrons_in_error_state.pl** and include any applicable usage parameters in the command line.

The software extracts the error state records as specified in your usage parameters and prints the output to the screen. If you want the data stored in a file, redirect the output into a file of your choosing. For example, to extract all patron records in error state into file **patron.mrc**, type:

```
extract_patrons_in_error_state.pl > patron.mrc
```

3.2.5 Extracting Patron Records by Date Range

The script **extract_patrons_by_modify_date.sh** creates a file of patron records that have been modified within a range of dates that you specify when running the script.

To run extract_patrons_by_modify_date.sh,

1. Log in to your server as **dbadmin**.
2. At the command prompt, type **extract_patrons_by_modify_date.sh**.
3. Press Enter.

   The script prompts you for a beginning and end date to define the date range.

4. Enter your preferred year, month, and day for the start and end dates, as prompted by the script.

Note: You can choose not to define an end date. If you do not choose an end date, **extract_patrons_by_modify_date.pl** creates a file of patrons whose records have been modified through the time you run the script.

The script extracts patron records that have been modified between the dates specified and writes them to a file of MARC records.

3.2.6 Extracting Duplicate Patron Records

The script **IdentifyDuplicatePatronBarcodes.sh** creates a list of duplicate patron barcodes in a database. This can be useful if your library uses both regular and alternate patron barcodes; e.g., in patron record fields 015 subfield $a and 016 subfield $a; as there may be instances where one patron's primary barcode is identical to another patron's alternate barcode.

To create a list of duplicate patron barcodes,

1. Log in to your server as **dbadmin**.
2. At the command prompt, type **`IdentifyDuplicatePatronBarcodes.sh`**.
3. Press Enter.

The script creates a comma-delimited file called **DuplicatePatronBarcodes.txt**. This file contains the patron_id, barcode number, and barcode type.

The barcode types are denoted as follows:
1 - main barcode (data comes from 015 tag)
2 - alternate barcode (data comes from 016 tag)
3 - hkid (data comes from 017 tag)
4 - Chamo Username (data comes from 014 tag)

3.2.7 Extracting Bibliographic Records for Discovery

If you use a third-party record service that accepts records in XML format only, you will find that the **ExtractForDiscovery.sh** script suits your needs. You can run the script to extract MARC 21 (UTF-8) bibliographic records in a Virtua database and convert them to XML format. The script first creates a file of bibliographic IDs of records that match the argument, which specifies a FULL extraction or an INCR(emental) extraction. Then the script automatically runs **write2709.exe** to extract the records and subsequently executes another program to convert the records to XML format.

3.2.7.1 Definition of a Full Extraction

A FULL extraction identifies all bibliographic records in the Virtua database that are not masked. When extracted, each MARC record (and subsequently the XML record) will contain 949 tags with associated item information for each item record attached to the bibliographic record. Any existing 949 tags will be removed and new 949 tags will be inserted.

The FULL extraction process generates two output files:

- **DiscoveryFullBib.rec** – The full set of MARC records.
- **DiscoveryFullBib.xml** – The XML version of the full set of MARC records.

3.2.7.2 Definition of an INCR(emental) Extraction

An INCR(emental) extraction identifies all bibliographic records in the Virtua database that 1) are not masked and 2) have been added or modified "x" number of days or greater, where "x" is the number entered on the command line. When extracted, each MARC record (and subsequently the XML record) will contain 949 tags with associated item information for each item record attached to the bibliographic record. Any existing 949 tags will be removed and new 949 tags will be inserted.

In addition, an INCR(emental) extraction identifies all bibliographic records in the Virtua database that have been deleted in the last "x" number of days. When extracted, no special 949 tag processing will be done on the MARC records (or subsequently on the XML records), but the record status in the Leader, position 05, of the extracted records will be set to 'd', indicating that they are deleted records. In this way, the target vendor will be able to identify the records to be deleted from the target system.

Finally, an INCR(emental) extraction identifies all bibliographic records in the Virtua database that are currently masked. The process extracts these records to a separate file and sets the status of the records (LDR/05) to 'd', indicating that these records should also be deleted from the target system.

The INCR(emental) extraction process generates six output files:

- **DiscoveryDeltBib.rec** – The deleted MARC records.
- **DiscoveryDeltBib.xml** – The XML version of the deleted MARC records.
- **DiscoveryIncrBib.rec** – The incremental MARC records.
- **DiscoveryIncrBib.xml** – The XML version of the incremental MARC records.
- **DiscoveryMaskBib.rec** – The masked MARC records.
- **DiscoveryMaskBib.xml** – The XML version of the masked MARC records.

Note: For the INCR(emental) extraction, the Virtua database currently does not support the generation of the date that a record was masked or unmasked.

3.2.7.3 Running ExtractForDiscovery.sh

To run ExtractForDiscovery.sh,

1. Log in to your server as **dbadmin**.

2. At the prompt, type: **`ExtractForDiscovery.sh [#]`**
   Where **#** represents the number of days as follows:

   - 0 – Indicates 0 number of days (i.e., Virtua will perform a FULL extraction).
   - Any number other than 0 – Indicates the number of days since the last extraction (i.e., Virtua will perform an INCR(emental) extraction).

   Examples:
   `ExtractForDiscovery.sh 0` < For a FULL extraction of bibliographic records.
   `ExtractForDiscovery.sh 14` < For an INCR(emental) extraction of bibliographic records that were added or modified in the LAST 14 days.

3. Press Enter.

4. Viewing Formatted MARC Records

Using the executable **MarcView.exe**, you can generate a file of formatted MARC records. Unlike **write2709.exe** which writes records in a format that is optimal for loading to the database, this program writes records in a format that is optimal for viewing. Below, we show the output of a record generated by the **MarcView.exe** program:

```
RecordId #802340:
LDR 00731 am  2200241I  4500
001 vtls000802340
003 VRT
005 20020722151700.0
008 020722 1970                00    pbk
020    $a 60221367
035    $a 0002-17960
039  9 $a 200207221517 $b staff $y 1999061516520000 $z load
090    $a SF AAPLA L6 2001
100    $a Avogadro, Lorenzo.
245 10 $a Protecting your Garden from Large Moles
260    $a Blacksburg $b Ten Twenty-third publications $c [2002]
300    $a 192 p.
440    $a Atomic Agriculture Publishing
500    $a Weighty subject matter
650  0 $a Agriculture
999    $a WORD
999    $a VIRTUA          m
999    $a
VTLSSORT0030*0050*0080*0200*0350*0390*0900*1000*2450*2600*3000*4400*
     5000*6500*9990*9991*9992
```

4.1 Running MarcView.exe

The **MarcView.exe** program accepts up to three optional command line options:

- **-s** - Specifies the character that will be used as the subfield delimiter in the output. If you do not include this option in the command, the program uses the default value of ****

- **-T** - Indicates the type of MARC record. Valid options are . . .
  - **101** - Bibliographic records
  - **102** - Authority records
  - **104** - Holdings records
  - **105** - Patron records

  If you do not include the **-T** option in the command, the program uses the default value of **101**.

- **-w** - Indicates the source of the records that will be extracted. Valid options are . . .
  - **0** - The Virtua database.
  - **1** - A special state table. If you choose this option, the program outputs the records you specify that are in the state table for the record type you indicate with the -T option.
  - **2** - A file of data. If you choose this option, specify the file of record data as the input file.

  If you do not include the **-w** option in the command, the program uses the default value of **0**.

Additionally, you need to specify an input file of record IDs and an output file to which **MarcView.exe** will write the formatted records.

To run MarcView.exe,

1. Log in to your server as **dbadmin**.
2. Type:

   ```
   MarcView.exe [command options] <[input filename] >[output
   filename]
   ```

   Where . . .

   - **[command options]** are the optional **-T**, **-s**, and **-w** options.

- **[input filename]** is the name of the file that contains the record IDs for the records you want to extract or, if you use the **-w2** option, the data file from which you want to extract records.
- **[output filename]** is the name of the file to which you want to write the extracted records.

  Note: If you specify the name of an existing file, the software will delete the contents of that file before writing data.

If, for a list of authority IDs in the file **auth_id.in**, you want to extract the corresponding records in the database to a file named **auth_rec.out**, using **$** as the subfield delimiter, type:

```
MarcView.exe -s$ -T102 -w0 <auth_id.in >auth_rec.out
```

3. Press Enter.

The program writes out the records to the specified file.

5. Cataloging/Processing Records

This chapter contains the following topics:

Virtua includes the following utilities for processing or reprocessing records:

- **MoveStateRecords.exe** - Reprocesses records of a specified state and moves them to a new state such as process.
- **ReProcessAuths.ksh** - Reprocesses the authority records specified in a file of record IDs.
- **ReProcessBibs.ksh** - Reprocesses the bibliographic records specified in a file of record IDs.
- **ReProcessFRBRWorksForSubjects.sh**- Indexes the subject headings of FRBR Work records
- **rlint.exe** - Processes records from RLIN so that they can be loaded to the Virtua database.
- **ReProcessForPublisherHeadings.sh** and **ReProcessForPublisherUserHeadings.sh** - Reprocesses the bibliographic records specified in a file of record IDs that contain 260$b or 264$b tags.

5.1 Reprocessing Records by State

The program **MoveStateRecords.exe** moves records from one state to another by reprocessing them. The program can also read load options from a file as long as the record state destination you choose is Catalog (i.e., Process Immediately). When

records are reprocessed, Virtua performs all the usual actions associated with saving the record, such as keyword and browse indexing.

Important:
- Use extra caution when using this script. Entering unintended responses to the prompts for information can cause the wrong records to be modified or even deleted. You can exit this program at any time by pressing the CTRL + C keys on your keyboard.
- Prior to running this program, cease all cataloging activity.

To run MoveStateRecords.exe,

1. Log in to your server as **dbadmin**.

2. At the prompt, type: **`MoveStateRecords.exe`**

   Notes:
   - You may use any command-line option with **MoveStateRecords.exe** that is available for **vload.exe**.
   - You can specify a file of load options to be used by **MoveStateRecords.exe**. Use the following format to specify a file of load options:
     `MoveStateRecords.exe -@[filename]`
     where **[filename]** is the name of the file of options.
   - For information on command-line options available for **vload.exe** and information on creating a file of load options, see the *Virtua Record Loading User's Guide*.

3. Press Enter.

   The program asks you to indicate which type of records you want to process.

4. Type:
   - **`a`** to process authority records.
   - **`r`** to process provisional authority records.
   - **`b`** to process bibliographic records.
   - **`h`** to process holdings records.
   - **`p`** to process patron records.

5. Press Enter.

The program asks you for the numeric ID of the state from which you want to move records and presents a list of states.

6. Type the numeric ID of the state from which you want to process records.
   -OR-
   Type **`all special`** to indicate all states *other than* **0** (i.e., Catalog).
   -OR-
   Type **`all states`** to indicate all states (except Deleted) *including* **0** (Catalog). If you choose this option, the program will try to catalog all records and move them to Catalog.

   Notes:
   - You cannot choose **0** (Catalog) at this prompt.
   - This program refers to the state Process Immediately as "Catalog."
   - All options except **all states** will include records in Deleted state.

7. Press Enter.

   If, in the previous step, you chose any option other than **all states**, the program asks you to type the numeric ID of the state to which you want to move records.

8. Type the ID of the state to which you want to move the records.
   -OR-
   If you want to delete records, type: **`delete`**.

   Notes:
   - Records in Error state will not be processed unless you choose **0** (Catalog) or **delete**.
   - If you choose to delete Error state records, base records will be retained. This means that for records that were previously in a normal state but were placed in Error state due to a modification, the record prior to the modification will be retained. Records that were put into Error state when they were first loaded or saved will be completely deleted.

9. Press Enter.

   The program prompts you to indicate which records you want to process.

10. Type the name of a file of record IDs that you want to process.
    -OR-
    Type **`all ids`** to process all qualifying records in the database.

11. Press Enter.

The program attempts to reprocess the specified records and move them to the state you specified. After processing is completed, the program exits.

5.1.1 Special Command-line Option for MoveStateRecords.exe

The **MoveStateRecords.exe** command-line option --do-not-update-authority-039-tag can be used with **MoveStateRecords.exe**, but NOT with **vload.exe**. This command-line option prevents Virtua from updating the 039 tag when **MoveStateRecords.exe** is run against Authority records.

Option: --do-not-update-authority-039-tag

Record Types: Authority

Description: Prevents Virtua from adding authority information to the 039 tag of incoming records.

Example:

```
MoveStateRecords.exe --do-not-update-authority-039-tag
```

Virtua will process the records according to other options without changing the 039 tag.

5.2 Reprocessing Authority Records

The script **ReProcessAuths.ksh** uses the program **MoveStateRecords.exe** to reprocess authority records in your database. By reprocessing each record, all actions associated with saving the record, such as browse indexing, are done for each record.

While reprocessing each authority record you specify, this script will attempt to move each record to the Process Immediately state (Catalog).

Important: Prior to running **ReProcessAuths.ksh**, cease all cataloging activity.

To run ReProcessAuths.ksh,

1. Log in to your server as **dbadmin**.

2. At the prompt, type: **`ReProcessAuths.ksh [file of IDs]`**
   Where **[file of IDs]** is the file of authority IDs that you want to reprocess.

3. Press Enter.

   The program attempts to reprocess the records you specified.

Note: ReProcessAuths.ksh is designed to reprocess permanent authority records, *not* provisional authority records.

5.3 Reprocessing Bibliographic Records

The script **ReProcessBibs.ksh** uses the program **MoveStateRecords.exe** to reprocess bibliographic records in your database. By reprocessing each record, all actions associated with saving the record, such as keyword indexing, are done for each record.

Note: When you modify bibliographic data directly by running a script or executable, the record will not be automatically re-keyword indexed via the keyword synchronization job, and the changes that you made will NOT be reflected in your keyword search results.

You have two server-side options for re-indexing modified bibliographic data:
--Running the script **ReProcessBibs.ksh** as described in this chapter.
-OR-
--Rebuilding the keyword index as describe in the *Virtua System Management: OPAC Parameters User's Guide.*

While reprocessing each bibliographic record you specify, this script will attempt to move each record to the Process Immediately state (Catalog).

Important: Prior to running this program, cease all cataloging activity.

To run ReProcessBibs.ksh,

- Log in to your server as **dbadmin**.

- At the prompt, type: **`ReProcessBibs.ksh [file of IDs] [log level]`**

  Where . . .
  - **[file of IDs]** is the file of bibliographic IDs that you want to reprocess.
  - **[log level]** is the level of logging you want to use. For this value, type:
    - 1 - Logs at debug level (the highest level of logging).
    - 2 - Logs at information level.
    - 3 - Logs warnings
    - 4 - Logs errors. If you do not specify a log-level, this level is used by default.
    - 5 - Logs only critical errors.
    - 6 - No logging

- Press Enter.

  The program attempts to reprocess the records you specified. Keep in mind that all records that are successfully processed will lose any special states that were previously assigned.

Note: If you set the environment variable ADD_AAP_LINKS_ONLY to **Y**, this script will index additional access points only. This means that the records will NOT be processed for keyword and call number indexing. For more information about this environment variable, see the *Virtua System Management Reference Guide*.

5.4 Reprocessing Patron Records

The script **ReProcessPatrons.sh** uses the program **MoveStateRecords.exe** to reprocess patron records in your database. By reprocessing each record, all actions associated with saving the record, such as browse indexing, are done for each record.

Note: When you modify patron name data directly by running a script or executable, the record will not be automatically re-indexed.

You have two server-side options for re-indexing modified patron name data:
--Running the script **ReProcessPatrons.sh** as described in this chapter.
-OR-
--Rebuilding the patron name browse index as describe in the *Virtua System Management: OPAC Parameters User's Guide*.

While reprocessing each patron record you specify, this script will attempt to move each record to the Process Immediately state (Catalog).

Important: Prior to running this program, cease all cataloging activity.

To run ReProcessPatrons.sh,

1. Log in to your server as **dbadmin**.
2. At the prompt, type: **`ReProcessPatrons.ksh [filename of patron IDs]`**

   Note: Consortium customers can run this script without any additional parameters.
3. Press Enter.

   The program attempts to reprocess the records you specified. Keep in mind that all records that are successfully processed will lose any special states that were previously assigned.

5.5 Reprocessing FRBR Records

If you have recently created FRBR records in your database, you can index the subject headings of the FRBR Work records without reindexing your entire database.

The script **ReProcessFRBRWorksForSubjects.sh** indexes the subject headings of FRBR Work records in your database.

To index Subject Headings of FRBR Work records in your database,

1. Log in to your server as **dbadmin**.
2. At the prompt, type:

   ```
   ReProcessFRBRWorksForSubjects.sh
   ```

3. Press Enter.

5.6 Processing Records from RLIN

Note: RLG/RLIN merged with OCLC in 2006.

If you obtain MARC records from RLIN, you cannot directly load these records to the Virtua database using **vload.exe**. Instead, you must first process these records to make them compatible with the Virtua data structure.

The executable **rlint.exe** prepares a file of records from RLIN for loading via **vload.exe**.

To process RLIN records so that they can be loaded to the Virtua database,

1. Log in to your server as **dbadmin**.
2. At the prompt, type:

   ```
   rlint.exe [input file] [output file]
   ```

   Where . . .

   - **[input file]** is the name (and, if necessary, the path) of the file of RLIN records.

- **[output file]** is the name of the file to which the program will write the processed records.

Tip: If you specify the name of an existing file, the software will delete the contents of that file before writing data.

For example, if the name of the input file is **RLIN.dat** and the name of the output file is **Virtua.dat**, type

```
rlint.exe RLIN.dat Virtua.dat
```

3. Press Enter.

5.7 Reprocessing Records for Publisher Headings

If you enable the setting *Index Publisher for Browse* in the Profiler's Cataloging Basic Options parameter, you will have to run a script in order for Virtua to index the publisher for browse searches. The following two scripts are associated with the *Index Publisher for Browse* setting:

ReProcessForPublisherHeadings.sh
ReProcessForPublisherUserHeadings.sh

Which script you run depends on whether you have previously indexed the 260 tag or 264 tag in bibliographic records as a user-defined browse search.

- If you have NOT previously indexed the 260 tag or 264 tag as a user-defined browse search, run **ReProcessForPublisherHeadings.sh**

- If you HAVE previously indexed the 260 tag or 264 tag as a user-defined browse search, run **ReProcessForPublisherUserHeadings.sh**

5.7.1 ReProcessForPublisherHeadings.sh

The **ReProcessForPublisherHeadings.sh** script creates a file of Bib IDs of records that contain the 260 subfield $b or 264 subfield $b. It reprocesses them to index the publisher headings only. It does not update the bibliographic records.

If you have NOT previously indexed the 260 tag or 264 tag as a user-defined browse search, run **ReProcessForPublisherHeadings.sh**. [If you have previously indexed those tags, run **ReProcessForPublisherUserHeadings.sh** instead (see below)].

When you run **ReProcessForPublisherHeadings.sh**, it will check for the following two conditions:

- That the *Index Publisher for Browse* setting in the Profiler's Cataloging Basic Options parameter is selected.
  -AND-
- That the 260 tag or 264 tag is not indexed as a user-defined browse search.

5.7.2 ReProcessForPublisherUserHeadings.sh

Note: Before you run **ReProcessForPublisherUserHeadings.sh**, you need to remove the 260 tag and 264 tag from the Tags Indexed for User Define Search parameter in the Virtua Profiler.

The **ReProcessForPublisherUserHeadings.sh** script creates a file of Bib IDs of records that contain the 260 subfield $b or 264 subfield $b. It reprocesses them to index the publisher headings AND user-defined headings, and it also updates the permanent authority records, if they exist. It does not update the bibliographic records.

If you have previously indexed the 260 tag or 264 tag as a user-defined browse search, run **ReProcessForPublisherUserHeadings.sh**. When you run the script, it will check for the following two conditions:

- That the *Index Publisher for Browse* setting in the Profiler's Cataloging Basic Options parameter is selected.
  -AND-
- That the 260 tag or 264 tag is *no longer* indexed as a user-defined browse search.

5.8 Reprocessing Bibliographic Records to Index VITAL PIDs

The script **ReProcessBibsToIndexVitalPid.sh** creates a file of Bib IDs that contain tag 856 subfield $i VITAL and reprocesses the Bib IDs to index the VITAL PID in the Vital_Pid column of the Bibliographic_Record table.

Note: This script is part of the functionality that supports the integration of VITAL and Virtua whereby VITAL digital content can be managed from within Virtua via the 856 tag in bibliographic records.

To run ReProcessBibsToIndexVitalPid.sh,

1. Log in to your server as **dbadmin**.
2. At the prompt, type: **`ReProcessBibsToIndexVitalPid.sh`**
3. Press Enter.

   The program locates and reprocesses the appropriate records.

5.9 Updating Bibliographic Record Fields

On rare occasions, after upgrading to a version of Virtua in which Innovative has made changes to the Virtua database structure, InfoStation reports may not run correctly, or some browse searches in the Virtua client will not return the correct results.

This can happen when certain parts of the new database structure have not been populated correctly. Normally, this will not be a problem. However, if you are having trouble with InfoStation or searches, and Innovative instructs you to do so, you can run the **PopulateBibFields.exe** executable to ensure any fields in the bibliographic_fields table are populated with bibliographic information.

Running **PopulateBibFields.exe** when it is not necessary will not do any harm to your database.

To run PopulateBibFields.exe,

1. Create a file of Bib IDs using WriteBibIdsFile.ksh.
2. Log in to your server as **dbadmin**.
3. At the database prompt, type

   `PopulateBibFields.exe <[filename]`

   Where **[filename]** is the name of the file of Bib IDs.
4. Press Enter.

The executable populates database tables used by InfoStation and browse searches.

5.10 Updating UDC Indexing

If you make any changes to the indexing rules that would affect UDC (Universal Decimal Code) browsing, you will need to run **PopulateUDCBrowse.sh** to apply these rules. This script will check if the database has UDC Browse tags define, and if so, it will call **PopulateUDCBrowse.exe** to create UDC browse entries for ALL bibliographic records based on the indexing rules.

Note: UDC tags are defined in the Virtua Profiler via the Cataloging tab > Tags Indexed for UDC Search.

To run PopulateUDCBrowse.sh,

1. Log in to your server as the **dbadmin** user.
2. Change directory to $EXE_DIR.
3. At the prompt, type:

   `PopulateUDCBrowse.sh`

The script runs:

- If no UDC tags are defined in the Virtua Profiler, the script displays **`No UDC Browse tags defined in the profiler`**.
- If UDC tags are defined in the Profiler, the script displays **`<Number of tags defined here> UDC Browse tag(s) have been identified, writing`**

`associated bib ids to a file...`, and sends the file of Bib-IDs to the **PopulateUDCBrowse.exe** program for processing

For information about UDC Browsing and altering the indexing rules, see the *Virtua Profiler/Cataloging Parameters User's Guide*.

5.11 Removing XML Data from Bibliographic Records

The script **RemoveXMLRecordFromBibs.sh** takes as input a file of bibliographic IDs and removes XML data that is part of those bibliographic records. The MARC data is left intact.

To remove XML data from bibliographic records,

1. Log in to your server as **dbadmin**.

2. At the database prompt, type:

   `RemoveXMLRecordFromBibs.sh [fileofbibIDs]`

Where **[fileofbibIDs]** is a file of bibliographic IDs in the database directory.

The script removes XML data from the specified bibliographic records in the database.

5.12 Unlocking MARC Records

The script **UnlockMARCRecord.sh** is designed to unlock a MARC record that is no longer being edited. It is meant to be used after you try to open a record in the Virtua client and receive the message "This record has been locked for updating."

Run this script only if you are certain that the record is not being edited and that the record is locked due to an error or crash in the client. If you are not certain of the status, you can run the script in "report-only" mode, which will not unlock the record.

Using the optional parameter [report_only], you can determine the date/time of the lock and the username of the person who locked it. In this way, you can judge whether a user may still be working inside of the record.

Usage: `UnlockMARCRecord.sh record_id record_type [report_only]`

Example: `UnlockMARCRecord.sh 50 105 report_only`

Where **50** is the ID of the record, **105** indicates that the record is a patron record, and **report_only** specifies that the script will report only without unlocking the record.

5.13 Adding a Special 035 Tag

Note: AddSpecial035.exe is for use by the following customers only: Thai Union Catalog, Université catholique de Louvain, National Library of Wales, and Hong Kong Public Library's Book Registration Office.

By running the program **AddSpecial035.exe**, you can add a special 035 tag to bibliographic records for use in duplicate checking. As an alternative, you can do the same thing by using the -3 option in **vload.exe**. For details about the executable and the load option, see the *Virtua Record Loading User's Guide*.

5.14 Populating the Control Number Index with Tags 024 and 028

By running the scripts **PopulateBibOtherIdentifier.sh** and **PopulateBibPublisherNumber.sh** you can populate the bib_other_identifier database table from bibliographic 024 tags and bib_publisher_number database table from bibliographic 028 tags. In this way, the Other Standard Identifier (024) and Publisher Number (028) will be available for selection in control number searches in the OPAC.

Indexing of the 024 tag and 028 tag is controlled by the following two settings in the Cataloging Basic Options parameter in the Virtua Profiler:

- Index Other Standard Identifier check box
- Index Publisher Number check box

Once you configure these settings, you need to run **PopulateBibOtherIdentifier.sh** and **PopulateBibPublisherNumber.sh** to populate the database tables. Each script needs to be run only one time.

To run PopulateBibOtherIdentifier.sh,

1. Log in to your server as **dbadmin**.
2. At the database prompt, type:

   `PopulateBibOtherIdentifier.sh`

3. Press Enter.

   The script populates the bib_other_identifier database table.

Use the same three steps to run **PopulateBibPublisherNumber.sh** to populate the bib_publisher_number database table.

6. Working with Authority Records

This chapter discusses the following topics:

6.1 Modifying Authority Records

Using the scripts **globalChange1.ksh** and **globalChange2.ksh**, you can find text in authority records stored in your database and replace that text with another value.

There are two steps for finding and replacing text in authority records:

1. Generate a list of authority IDs for authority records containing the text you want to replace.
2. Find and replace text within a tag range for the file of authority IDs you specify.

6.1.1 Generating a List of Authority IDs Containing Specified Text

The script **globalChange1.ksh** generates a list of authority IDs for the authority records containing the text that you specify.

To use globalChange1.ksh to generate a list of authority IDs,

1. Log in to your server as **dbadmin**.

2. At the prompt, type:

```
globalChange1.ksh [authority type] [text to find]
```

Where . . .

- **[authority type]** is one of the following one-character codes:
  - **x** - Author/name
  - **y** - Subject
  - **z** - User-defined
  - **t** - Series title
  - **u** - Uniform title
  - **d** - Subdivision
  - **w** - Genre/form
- **[text to find]** is the text that must appear in the authority record for the authority ID to be extracted.

3. Press Enter.

The script writes to a file named **authIDs.dat** a list of authority IDs that meet the criteria you specified.

6.1.2 Replacing Text in Authority Records

After you have used **globalChange1.ksh** to generate a list of authority IDs, you can use **globalChange2.ksh** to replace text in those records.

To use globalChange2.ksh to replace text in authority records,

1. Log in to your server as **dbadmin**.

2. At the prompt, type:

```
globalChange2.ksh [tag range] [old text] [new text] [filename]
```

Where . . .

- **[tag range]** is one of the following one-digit codes:
  - **1** - 1XX tags
  - **4** - 4XX tags
  - **5** - 5XX tags
- **[old text]** is the text occurring in the tag range you specified that you want to replace.
- **[new text]** is the text with which you want to replace the old text.

- **[filename]** is the name of the file that lists the authority IDs of the records that this script will check.

3. Press Enter.

The script replaces the text in the authority records you specified.

6.2 Adding a Field to Authority Records to Avoid Conflict

The script **FixAuthConflictErrors.sh** prevents 4xx conflicts in authority records with the defined authority control number prefix by adding a subfield $z to the 4xx tag of each of the records in conflict. The new subfield $z will contain the 035 control number.

This script is useful when authority records have been ingested that caused a conflict with existing authority records, and this resulted in a number of authority records that are in Error state.

To run FixAuthConflictErrors.sh,

1. Log in to your server as **dbadmin**.

2. At the prompt, type:

   ```
   FixAuthConflictErrors.sh [prefix]
   ```

   Where **[prefix]** is the prefix of the authority control number to use. If no prefix is defined, then ALL authority records will be processed.

3. Press Enter.

Virtua modifies the authority records with the specified prefix to add to the 4xx tag a subfield $z that will contain the contents of the 035 tag.

6.3 Changing Permanent Authority Records to Provisional

You can convert permanent authority records that have no cross-references or control number tags to provisional authority records. To do this, use the scripts **PermAuthIdsToProv.ksh** and **ChangePermAuthorityToProv.sh**.

There are two steps for converting authority records:

1. Create a file of IDs for permanent authority records that are candidates for conversion back to provisional records. Records that qualify have a 1XX tag but no cross-references (4XX, 5XX, or 7XX tags) and no control numbers (010 or 035 tags).
2. Change the permanent records to provisional records.

6.3.1 Generating a List of Candidate Records

To generate a list of candidates for conversion to provisional records,

1. Log in to your server as **dbadmin**.
2. At the prompt, type:

   ```
   PermAuthIdsToProv.ksh [ID file]
   ```

   Where **[ID file]** is the name of the file to which the program will write the IDs of records that are candidates for conversion to provisional records.

3. Press Enter.

The program finds all permanent authority records that have a 1XX tag but no cross-references (4XX, 5XX, or 7XX) and no control numbers (010 or 035 tags). The program creates two files:

1. A file of authority IDs for the records that qualify for conversion. The name of this file is determined by the file you specify as a command option when you run the script.

2. A file of the records, in a viewable format, that are associated with the record IDs. The name of this file is the same as the file of IDs generated by the program with an extension of **.print**. For example, if the name of the file of IDs is **prov_auths**, the name of the file of records is **prov_auths.print**.

After the program generates the files, review the file of generated records to make sure that you want to convert them to provisional records. For any records in the file that you do not want to convert, remove the associated authority ID from the file of IDs.

6.3.2 Changing to Provisional Records

To convert permanent authority records to provisional records,

- At the prompt, type:

  ```
  ChangePermAuthorityToProv.sh [ID file]
  ```

  Where **[ID file]** is the name of the file that contains a list of IDs of permanent authority records that you want to convert to provisional. This is usually the file created by running **PermAuthIdsToProv.ksh**.

- Press Enter.

The program converts the specified authority records to provisional authority records. Authority records that are not linked to a bibliographic record or a blind reference are ignored.

6.4 Removing Orphaned Authority Headings

The script **authClr.sh** removes orphaned authority headings and orphaned bibliographic IDs from the browse list. Orphaned headings are headings that do not have attached bibliographic records and are not blind references. Orphaned bibliographic IDs are those IDs that remain in the database although they have no associated MARC bibliographic data.

> **Important:** Prior to running this program, cease all cataloging activity.

To run authClr.sh,

1. Log in to your server as **dbadmin**.
2. At the prompt, type: **`authClr.sh`**
3. Press Enter.

   The program finds and removes orphaned authority headings and orphaned bibliographic IDs from the browse list.

6.5 Locating "Un-linked" 5xx Headings

You can run the script **CheckForReciprocal1xx5xxAuthorities.sh** against the database to check for "un-linked" 5xx cross-references. An un-linked 5xx cross-reference is a 5xx tag *See also* heading that is not linked to a 1xx tag in an LC authority record. In other words, the script checks for 5xx tags that do not also appear in their "own" authority record as a 1xx.

Running the script **CheckForReciprocal1xx5xxAuthorities.sh** creates an output file that lists the IDs of authority records that have been found to have "un-linked" 5xx headings. The output file contains the following information about each record found:

- Authority bib lvl
- Auth ID
- Brief display of the "un-linked" 5xx heading

To run CheckForReciprocal1xx5xxAuthorities.sh,

1. Log in to your server as **dbadmin**.
2. At the prompt, type:

```
CheckForReciprocal1xx5xxAuthorities.sh <startDate> <endDate>
```

Where the date range reflects the modification date of the authority records that will be examined.

The software creates the output file called **CheckForReciprocal1xx5xxAuthorities.[time stamp].log**, where **[time stamp]** represents a time stamp of the form DD_MM_YYYY_HH_MM_SS for the year, month, day, hour, minute, and second when the script was run.

6.6 Extracting Permanent Authority Records to a File

The script **WritePermAuthorityFile.ksh** uses **write2709.exe** to write to a file every *permanent* authority record in the database.

To extract a file of permanent authority records,

1. Log in to your server as **dbadmin**.
2. Type: **`WritePermAuthorityFile.ksh [output filename]`**

   Where **[output filename]** is the name of the file to which you want to write the permanent authority records.

   Note: If you specify the name of an existing file, the software will delete the contents of that file before writing authority records.

3. Press Enter.

   The script writes out the authority records to the specified file.

6.7 Extracting Authority Records from a File of Authority Control Numbers

The script **ExtractAuthorityRecordsUsingControlNumbers.sh** connects to a Virtua database and creates a file of authority records from a file of authority control numbers.

To extract a file of authority records from a file of authority control numbers,

1. Log in to your server as **dbadmin**.
2. Type: **`ExtractAuthorityRecordsUsingControlNumbers.sh < name of the file of authority control numbers`**
3. Press Enter.

   The script writes out the authority records to **AuthsFromControlNumbers.rec**.

7. Modifying Bibliographic Data Using MARCBibUpdate.exe

The program **MARCBibUpdate.exe** lets you add, modify, and delete data in MARC records. With this utility, you can alter, in batch, bibliographic records in a database or any type of MARC records in a file. Because this program is one that irrevocably changes data in bibliographic records in the database, do NOT run **MARCBibUpdate.exe** unless you are certain about what you are doing. Read the entire contents of this chapter before running this program.

Important: Keep in mind the following recommendations for running **MARCBibUpdate.exe**:

- If you choose to have the program directly modify data in the database, keep in mind that it will bypass the safeguards in the Virtua program server that help prevent record corruption. It is important that you make certain that the changes the program makes will not corrupt records.
- If you must modify records in the database, take an export of the Bibliographic_Record table. If running **MARCBibUpdate.exe** does not produce the desired results, you can use the export to return the database to the previous state. If you do not have an export of this table, *you will not be able to undo any modifications made by this program*.
- If you need to use this program to make changes to indexed data, read the section "Modifying Indexed Data" in this chapter.
- Do NOT use this program to add, delete, or modify tags that are included in authority records. If you need to make changes to tags that are included in authority records, modify the authority record in Virtua, or use the scripts **globalChange1.ksh** and **globalChange2.ksh** to make batch modifications of authority records.

In this chapter we discuss the following topics:

7.1 Command Line Reference for MARCBibUpdate.exe

This section describes the command line options for **MARCBibUpdate.exe**.

Important: We recommend that you read and understand ALL the information in this section before you run **MARCBibUpdate.exe**.

You can run **MARCBibUpdate.exe** in the following format:

MARCBibUpdate.exe [file option] [selection block] [action block]

Where . . .

- **[file option]** specifies a file of records or IDs that the program will use.
- **[selection block]** defines the attributes of the records that will be modified.
- **[action block]** determines the modifications that will be made to the records that meet the criteria for selection.

To get online help regarding options to be used with **MARCBibUpdate.exe**, type:

```
MARCBibUpdate.exe -help
-OR-
MARCBibUpdate.exe -?
```

Note: MARCBibUpdate.exe does not handle input strings with diacritics.

7.1.1 Understanding Selection Blocks

A selection block determines the criteria that a record must meet to be selected by **MARCBibUpdate.exe**. If a record does not meet the criteria defined in the selection block, it will not be affected by the options defined in an associated action block.

A selection block usually begins with an option that determines the tag for which the following options will apply. All subsequent options in the selection block refer to the defined tag until a new tag number is defined. A selection block ends at the first option in an action block or when the --reverse-select option is specified.

7.1.2 Understanding Action Blocks

An action block specifies the action that the program will take on the record. In most cases, the action will add, delete, or modify information in the last tag specified by the selection block. An action block must be preceded by a selection block.

When an action block specifies that an addition, deletion, or modification is to be made, it will be done on the last tag or field defined in the associated selection block.

7.1.3 Using Multiple Selection and Action Pairs

If desired, you can define multiple selection and action pairs by using the following format:

MARCBibUpdate.exe [selection 1] [action 1] --and-select [selection 2] [action 2]

If you use this format, the records that will be modified must qualify for both selection blocks. If a record fails to meet one of the selection blocks, it will not qualify for either action.

Note:

- Records that qualify for both selection blocks will be modified by the first action before it is modified by the second action. The second action applies to the records modified by the first action.
- If the *--and-select* is used for the sole purpose of further qualifying a selection, the option *--no-action* can be used to indicate the end of the extra qualifications.

7.1.4 Specifying Options in a File

Since the options for **MARCBibUpdate.exe** can exceed the maximum number of characters allowed on the command line, we recommend that you specify options in a file. To do this, enter the options in a text file as you would type them on a command line.

For readability, you can include line breaks in your file of options. The program will interpret each line break as a single space. For example:

--tag-number 555 --subfield-code a --subfield-exactly 'Includes bibliographical references and indexes.' --change-tag-number 504

When you run the program, call the file by using the following format:

MARCBibUpdate.exe --input [filename]

Where **[filename]** is the name of the file containing command options.

Note: If you use the --input option, it must be the only option in the command line. All other options must be specified in the file itself.

7.1.5 Choosing a File Option

This section lists and describes each of the file options that you can use with **MARCBibUpdate.exe**.

7.1.5.1 Record File

Option: --record-file

Description: This parameter specifies the name of a file of MARC records to be examined. The file of records must be in the directory in which you are running **MARCBibUpdate.exe**, or you must specify a relative path to the file. If you use this option in the command, the program will NOT access or modify data in the database.

Option value: The name of a file of MARC records.

Example: --record-file holdings.rec

Note:
- If you use this option in your command, you must also include the --write-all or --write-changed options.
- If you use this option in your command for records *other than bibliographic records*, you must also specify a file as output.
- Do NOT use this option with the --id-file option or if you are making changes to the 003 tag.

7.1.5.2 Update File

Option: --update-file

Description: This parameter specifies the name of a file of records to be examined and updated. Using the bib-id from the 001 tag, it replaces the record in the database. The file of records must be in the directory in which you are running **MARCBibUpdate.exe**, or you must specify a relative path to the file. If you use this option in the command, the program will not access or modify data in the database.

Option value: The name of a file of bibliographic records to be updated.

Example: --update-file bibliographic.rec

7.1.5.3 File of IDs

Option: --id-file

Description: This parameter specifies the name of a file of bibliographic record IDs. This file determines which records in the database are examined by the program.

Option value: The name of a file of bibliographic record IDs.

Example: --id-file bibliographic.ids

Note: Do NOT use this option with the --record-file option.

7.1.6 Choosing Selection Options

This section describes each of the selection options that you can use with **MARCBibUpdate.exe**. Selection options determine the criteria that a record must meet to be modified by the program.

A block of selection options must be followed by one or more options that specify an action to perform on the qualifying records. For information about options for specifying an action, see the section "Choosing Action Options" in this user's guide.

Note:
- All values that you specify for selection options are case sensitive.
- If, for an option, you specify a value that includes one or more spaces, enclose the value in quotation marks.

7.1.6.1 Tag Number Definition

Option: --tag-number

Description: This option specifies one or more tag numbers for the selection block.

Option value: One or more tag number definitions. You can substitute an **X** to indicate a wildcard (ex. 5**XX**). To specify multiple tags, separate each tag with a single space.

Example: --tag-number 110

Note: All subsequent options in the selection block will refer to the tags you define in this option.

7.1.6.2 Tag Occurrence

Option: --tag-occurrence

Description: This option specifies which occurrence of the tag specified by the --tag-number option is used in the selection.

Option value: A number that specifies which occurrence of the tag you want to select.

Example: --tag-occurrence 1

Note: If you want to select all occurrences of the tag, do not include this option in the command.

7.1.6.3 First Indicator

Option: --first-indicator

Description: This option restricts the selection by specifying a value that must exist in the first indicator of the tag. If the tag does not have the specified value in the first indicator, it is not a match.

Option value: An indicator value.

Example: --first-indicator 8

Note: You CANNOT use this option when the last --tag-number option specifies a fixed-field tag.

7.1.6.4 Second Indicator

Option: --second-indicator

Description: This option restricts the selection by specifying a value that must exist in the second indicator of the tag. If the tag does not have the specified value in the second indicator, it is not a match.

Option value: An indicator value.

Example: --second-indicator 2

Note: You CANNOT use this option when the last --tag-number option specifies a fixed-field tag.

7.1.6.5 Subfield Code

Option: --subfield-code

Description: This option specifies one or more subfields that might exist in the selected tag.

Option value: One or more subfield codes. To specify multiple subfield codes, type each code without separating with spaces (ex. **abc**).

Example: --subfield-code a

Note: You CANNOT use this option when the --tag-number option specifies a fixed-field tag.

7.1.6.6 Subfield Occurrence

Option: --subfield-occurrence

Description: This option specifies which occurrence of the subfield specified by the --subfield-code option is used in the selection.

Option value: A number that specifies which occurrence of the subfield you want to select.

Example: --subfield-occurrence 1

Note: If you want to select all occurrences of the subfield, do not include this option in the command.

7.1.6.7 Subfield Value (Exact Match)

Option: --subfield-exactly

Description: This option specifies a string that must match exactly the value in the selected subfield.

Option value: The value that must exist in the selected subfield.

Example: --subfield-exactly 'Blacksburg, VA'

7.1.6.8 Subfield Value (Substring)

Option: --subfield-substring

Description: This option specifies a string that must occur in the subfield specified by the --subfield-code option used in the selection. While the string does not have to match the *entire* subfield value, it must exist in the field.

Option value: The value that must exist in the selected subfield.

Example: --subfield-substring Blacksburg

7.1.6.9 Subfield Value (Fixed Position)

Option: --subfield-fixed

Description: This option specifies a fixed position in the subfield specified by the --subfield-code option and a value that must begin at that position.

Option values: This option accepts two values:

- A number that specifies the offset in the subfield at which the value begins.

-AND-

- The value that must occur from the offset position.

Example: --subfield-fixed 21 a

Hint: This option is useful for finding values in hybrid variable/fixed field tags such as the 999 VIRTUA tag.

7.1.6.10 Subfield Range Value

Option: --subfield-range

Description: This option specifies a range of values in the subfield specified by the --subfield-code option.

Option values: This option accepts two values:

- The first value in the range.

  -AND-

- The last value in the range.

Note: The first value and last value must contain the same number of characters.

For a record to be selected, the subfield must contain a value between the specified first and last values, according to standard text ordering.

Example: --subfield-range 200301011200 200401011200

Hint: This option is useful for specifying a date range.

7.1.6.11 Fixed Field Value

Option: --fixed-tag

Description: This option specifies a position in the fixed field tag specified by the --tag-number option and a value that must begin at that position.

Option value: This option accepts two values:

- A number that specifies the offset in the tag at which the value begins.
  -AND-
- The value that must occur from the offset position.

Example: --fixed-tag 35 epo

Note: You CANNOT use this option when the last --tag-number option specifies a variable field tag.

7.1.6.12 Leader Value

Option: --leader-string

Description: This option specifies a position in the record leader and a value that must begin at that position.

Option value: This option accepts two values:

- A number that specifies the offset in the record leader at which the value begins.
  -AND-
- The value that must occur from the offset position.

Example: --leader-string 18 i

Note: This option does not require any other selection option to be included in the command.

7.1.6.13 Capturing Subfield Data

Option: --capture-subfields

Description: This option specifies that the specified subfields from the selected variable field tag will be captured and held in memory. The captured data can then be inserted into a variable field tag with the --insert-once-per-capture option.

Option value: One or more subfield codes for the data that you want to capture. To specify multiple subfield codes, type each code without separating with spaces (ex. **abc**).

Example: --capture-subfields af

7.1.6.14 Capturing Fixed Field Data

Option: --capture-fixed-tags

Description: This option specifies that the fixed-field tag specified in the selected fixed field tag will be captured and held in memory. The captured data can then be inserted into a variable field tag with the --insert-once-per-capture option.

Option value: None.

7.1.6.15 Comparing by Using Normalized Versions

Option: --compare-normalized

Description: This option modifies the options *--subfield-fixed*, *--subfield-exactly*, *--subfield-string*, and *--fixed-tag* by using normalized (non-case-sensitive) versions when making searches for a substring. Normalized versions of both the substring and the data being searched for the substring are used. This option modification applies only to those search options that follow the first appearance of *--compare-normalized*.

Option value: None.

7.1.6.16 Blocking a Compare-Normalized

Option: --compare-unchanged

Description: This option blocks the effects of a *--compare-normalized* on search options that follow it. The *--compare-normalized* searching applies only to those options that are preceded by a *--compare-normalized* option without an intervening *--compare-unchanged*.

Option value: None.

7.1.6.17 Excluding Records that Match the Selection Criteria

Option: --reverse-select

Description: This option specifies that the criteria in the selection block will be used to *exclude* the matching records from being modified. The actions defined in the action block will be done on the records that do NOT meet the criteria defined in a selection block that uses this option.

Option value: None.

7.1.7 Choosing Action Options

This section describes each of the action options that you can use with **MARCBibUpdate.exe**. Action options determine what action is done for the records that meet the criteria specified in the selection block.

The options listed in this section vary widely in their actions. Before using an option, read each description carefully.

Hint: If, for an option, you specify a value that includes one or more spaces, enclose the value in quotation marks.

7.1.7.1 Writing All Records to a File

Option: --write-all

Description: This option specifies that the program writes to a file all records that it examines.

Option value: The name of the file to which the program will write records.

Example: --write-all bibliographic.rec

Note: If your selection block includes the --record-file option, you must include this option or the --write-changed option.

7.1.7.2 Writing Modified Records to a File

Option: --write-changed

Description: This option specifies that the program writes to a file all records that it modifies. Records that do not qualify for the criteria in the selection block will not be included in the file.

Option value: The name of the file to which the program will write modified records.

Example: --write-changed modified_bibliographic.rec

Note: If your selection block includes the --record-file option, you must include this option or the --write-all option.

7.1.7.3 Archiving Records

Option: --archive-record

Description: This option archives the record before an update is made.

Option value: None

7.1.7.4 Restoring Records from Archive

Option: --restore-from-archive

Description: This option overwrites the record with the archived record from the specified date range. Records archived before the early date or after the late date will not be considered. When comparing the archive date to the early or late date, the archive date is truncated to the same precision provided in the early date or late date.

- If only one date is given, the truncated version of the archive date must match the date provided.
- If multiple archive records for a single bib-id exist in the date range specified, an error message will be generated, and the restore will be blocked.
- If no date is given, the most recent archive record with the same bib-id will be used.

Option values: This option accepts two values:

- An early date in the format of YYYYMMDDHH24MISS (or a subset of this format)
- A late date in the format of YYYYMMDDHH24MISS (or a subset of this format)

Example: --restore-from-archive 19990101 19991231

7.1.7.5 Printing the ID of Selected Records

Option: --print-id

Description: This option specifies that the program write an ID for each record that meets the criteria specified in the selection block. If the records are . . .

- From the database, the ID is the bibliographic ID (001 tag).
- From a file of records, the ID is a number representing the position of the record in the file. For example, the first record's ID is **1**, the fifth record's ID is **5**, and the tenth record's ID is **10**.

Option value: The name of the file to which the program will write IDs.

Example: --print-id selected.ids

7.1.7.6 Adding a Tag

Option: --add-tag

Description: This option adds a tag to selected records.

Option value: This option accepts three values:
- The three-digit number of the tag you want to add.
- The entire text of the tag, including indicators and subfield codes.
- An optional parameter that specifies the occurrence of the tag. If you do not specify an occurrence, the tag will be inserted as the last occurrence.

Rules for defining the tag data:
- To specify a blank indicator, enter a single space.
- Use the $ character as a subfield delimiter.
- A subfield delimiter must follow immediately after the second indicator definition.
- For subfields defined after the first subfield in the tag, the delimiter should follow immediately after the last character in the preceding subfield data.
- To enter a dollar sign ($) that will not be used as a subfield delimiter, type **$$**.

Example: --add-tag 521 '3 $3 Large Print$aVisually Impaired'

7.1.7.7 Adding a Subfield

Option: --add-subfield

Description: This option adds a subfield to the tag defined by the last --tag-number option in the selection block.

Option value: This option accepts three values:
- The subfield code for the subfield you want to add.
- The entire text of the subfield.
- An optional parameter that specifies the occurrence of the subfield. If you do not specify an occurrence, the subfield will be inserted as the last occurrence.

Example: --add-subfield d 'Faculty Access Only'

7.1.7.8 Deleting the Selected Tag

Option: --delete-tag

Description: This option deletes the tag defined by the last --tag-number option in the selection block.

Option value: None.

7.1.7.9 Deleting the Selected Subfield

Option: --delete-subfield

Description: This option deletes the subfield defined by the last --subfield-code option in the selection block.

Option value: None.

7.1.7.10 Changing the Selected Tag Number

Option: --change-tag-number

Description: This option changes the tag indicated by the last --tag-number option in the selection block to a new number.

Option value: A three-digit tag number.

Example: --change-tag-number 110

This example changes the tag number of the tag defined by the --tag-number option in the selection block to 110.

7.1.7.11 Changing the Selected Subfield Code

Option: --change-subfield-code

Description: This option changes the subfield code defined by the last --subfield-code option in the selection block to a new code.

Option value: A subfield code.

Example: --change-subfield-code a

This example changes the subfield code of the subfield defined by the --subfield-code option in the selection block to **a**.

7.1.7.12 Replacing the Leader String

Option: --replace-leader-string

Description: This option replaces the portion of the record leader that matches the criteria specified by the last --leader-string option in the selection block.

Option value: One or more characters to replace the portion of the leader specified by the --leader-string option. The length of the value must match exactly the length of the replaced string.

Example: --replace-leader-string 1

Note: You cannot replace values in the following ranges of positions in the record leader:

- 0-4
- 10-16
- 20-23

7.1.7.13 Replacing a String in a Fixed Field Tag

Option: --replace-tag-string

Description: This option specifies the text that will replace the string defined by the last --fixed-tag option defined in the selection block for a fixed field tag.

Option value: One or more characters to replace the portion of a fixed field tag specified by the --fixed-tag option. The length of the value must match exactly the length of the replaced string.

Example: --replace-tag-string a

7.1.7.14 Replacing a String in a Subfield (For the Entire Subfield)

Option: --replace-subfield

Description: This option specifies the text that will replace the value of the subfield specified by the last --subfield-code option in the selection block.

Option value: The text that you want to use to replace the value of the subfield. This text will replace the *entire* value of the subfield.

Example: --replace-subfield 'Faulkner, William'

7.1.7.15 Replacing a String in a Subfield (For a Defined Substring)

Option: --replace-subfield-string

Description: This option specifies the text that will replace the text defined by the --subfield-substring option in the subfield defined by the --subfield-code option.

Option value: The text that you want to use to replace the value of the string specified by the --subfield-substring option.

Example: --replace-subfield-string Faulkner

Note:
- If there is more than one --subfield-code option in the selection block, the last one defined will be the one that determines which subfield is used.
- If there is more than one --subfield-substring option in the selection block, the last one defined will be the one that determines which text is replaced.

7.1.7.16 Replacing a Tag

Option: --replace-tag

Description: This option specifies the value that will replace the tag defined by the last --tag-number option in the selection block.

Option value: The entire text of the tag, including indicators and subfield codes.

Rules for defining the tag data:
- To specify a blank indicator, enter a single space.
- Use the $ character as a subfield delimiter.
- A subfield delimiter must follow immediately after the second indicator definition.
- For subfields defined after the first subfield in the tag, the delimiter should follow immediately after the last character in the preceding subfield data.
- To enter a dollar sign ($) that will not be used as a subfield delimiter, type **$$**.

Example: --replace-tag '10$a Flags in the dust.$c Edited by Douglas Day.'

7.1.7.17 Setting the First Indicator Value

Option: --set-first-indicator

Description: Sets the value of the first indicator for the tag specified by the last --tag-number option in the selection block. If the tag already has a first indicator value, it will be replaced by the new value.

Option Value: The value to which the first indicator will be set.

Example: --set-first-indicator 4

7.1.7.18 Setting the Second Indicator Value

Option: --set-second-indicator

Description: Sets the value of the second indicator for the tag specified by the last --tag-number option in the selection block. If the tag already has a second indicator value, it will be replaced by the new value.

Option Value: The value to which the second indicator will be set.

Example: --set-second-indicator 1

7.1.7.19 Setting the Value of Part of a Subfield

Option: --set-subfield-part

Description: Sets the value from a specified position in the subfield defined by the last --subfield-code option in the selection block. The program will overwrite any data that exists in the location at which you specify the data be written.

Option Value: This option accepts two values:

- A number that specifies the offset in the tag at which the text is added.
- The text to add.

Example: --set-subfield-part 6 1934

7.1.7.21 Inserting a Value in a Subfield

Option: --insert-subfield-part

Description: Inserts text in the specified position of the subfield defined by the last --subfield-code option in the selection block. This option differs from the --set-subfield-part option in that it does not overwrite data.

Option Value: This option accepts two values:

- A number that specifies the offset in the tag at which the text is inserted.
- The text to insert.

Example: --insert-subfield-part 6 1997

7.1.7.22 Inserting a Captured Value

Option: --insert-once-per-capture

Description: Inserts text captured by the --capture-subfields or --capture-fixed-tags options in the specified tag or subfield specified by the --add-tag or --add-subfield options. The captured text replaces the specified insertion marker in the tag or subfield. If the tag or subfield does not include the insertion marker, the program will not insert the captured text.

Option Value: An insertion marker that exists in the tag or subfield. When you add the tag or subfield, you should include the insertion marker in the new data.

Example: --insert-once-per-capture #

Note:

- For each tag or subfield that qualified for capture, a new tag or subfield will be added.
- If there are multiple insertion markers in the tag or subfield, the captured data will be inserted only at the first marker.

7.1.7.23 Blocking Updates to the bibliographic_fields Table

Option: --block-update-of-bib-fields

Description: Blocks the update of the bibliographic_fields table.

Option Value: None

Note: This option is used to speed up the change process when changing fields in the bibliographic record that are NOT in the bibliographic_fields table.

7.2 About Modifying Indexed Data

Note: If you are modifying indexed data, we strongly recommend that you use the workflow outlined in the section "Recommended Workflow" in this chapter. By using this workflow, you can ensure that all indexing is done for the records you modify.

If you are directly modifying records in the database and you need to make changes to tags that are indexed for OPAC searches, keep in mind that **MARCBibUpdate.exe** will not update the search index. If you use this program to modify indexed data, you will need to make sure that the modified fields are indexed

If you modified a field that is indexed for . . .

- Bibliographic keyword searching, you can run **KeywordIndex.exe** to update the table of keyword data. For information about running **KeywordIndex.exe**, see the *System Management: OPAC User's Guide*.
- Call number browsing, you can run **CallIndexForm.sh** to update the call number index. For information about running **CallIndexForm.sh**, see the *System Management: OPAC User's Guide*.
- Universal Decimal Classification control number browsing, you can run **PopulateUDCBrowse.exe**. For information on this script, see the section "Updating UDC Indexing" in this user's guide.
- If you modified a field that is indexed for any other kind of search, such as a control number, you will need to reprocess the record. You can use the script **ReProcessBibs.ksh** to reprocess bibliographic records. For information on this script, see the section "Reprocessing Bibliographic Records" in this user's guide.

7.3 Recommended Workflow

Important: Do NOT use this workflow if you are using **MARCBibUpdate.exe** to modify the 003 tag. If you are changing the value of the 003 tag, you must make this change directly on the database. We strongly recommend that you take an export of the Bibliographic_Record table before running this program.

This section describes a recommended workflow for running **MARCBibUpdate.exe**. This workflow lets you extract records from your database for modification so that you can view the changes that were made before you load the modified records to the database.

This workflow offers the following advantages:

- You have the opportunity to verify the appropriate changes have been made before records in your database are modified.
- Virtua checks the basic record structure as each record is reloaded.
- All indexing processes are done as each record is reloaded.

Note: If this workflow does not fit your needs, we still recommend that you follow it to test your changes on a selection of records.

To run MARCBibUpdate.exe,

1. Log in to your server as **dbadmin**.

2. Make sure that the ORACLE_SID and EXE_DIR environment variables are set correctly. For information about setting these variables, see the section "An Important Note About Environment Variables" in this user's guide.

3. Navigate to a directory to which you can write files.

4. If you have available a file of record IDs on which you want to make modifications, copy this file to the current directory.

5. Create a file of options for the modifications that you want to make. Include in the action block the --write-changed option. This option instructs the program to write to a file the records that are modified by **MARCBibUpdate.exe**. No changes are made to the database.

6. At the prompt, type:

   ```
   MARCBibUpdate.exe --input [filename]
   ```

   Where **[filename]** is the name of the file of options that you created in the previous step.

7. Press Enter

   The program finds the records that qualify for selection and writes the modified records to the a file you specified in the --write-changed option.

8. Use the program **MarcView.exe** to view the file of records. For information about running **MarcView.exe** see the chapter "Viewing Formatted MARC Records" in this user's guide.

When viewing records, make sure that the appropriate changes have been made. If there are unwanted changes to the records, do NOT continue. Instead, check the options that **MARCBibUpdate.exe** used and start this process again. If you still do not get the desired results and do not know why, contact Innovative customer services for assistance.

9. If you are certain that the appropriate changes have been made, use **vload.exe** to load the records to your database. For information on running vload.exe, see the *Virtua Record Loading User's Guide*.

   By using **vload.exe** to load the records, you can make sure that all indexing processes are run on the records. Additionally, since **vload.exe** uses the Virtua cataloging process, there is some degree of validation done on each record.

7.4 Examples

In this section we provide several examples of command options for running **MARCBibUpdate.exe**. For details about the options used in these examples, see the section "Command Line Reference for MARCBibUpdate.exe" in this user's guide.

7.4.1 Adding a Tag

The following set of options instructs **MARCBibUpdate.exe** to . . .

- Select all records with a value of *e* in position 22 of the 008 tag.
- Add a 521 tag with . . .
  - A first indicator value of *1*
  - A subfield $a with a value of *Ages 18 and older*.

Options:

--tag-number 008 --fixed-tag 22 e --add-tag 521 "1 $aAges 18 and older"

7.4.2 Changing a Tag Number

The following set of options instructs **MARCBibUpdate.exe** to . . .

- Select all records with an exact value of *Young Adult* in tag 500 subfield $a.
- Change the tag number of the selected tag to 521.

Options:

--tag-number 500 --subfield-code a --subfield-exactly "Young Adult" --change-tag-number 521

7.4.3 Deleting a Tag

The following set of options instructs **MARCBibUpdate.exe** to . . .

- Select all records with an exact value of *Faculty and Staff* in tag 506 subfield $d.
- Delete the selected tag.

Options:

--tag-number 506 --subfield-code d --subfield-exactly "Faculty and Staff" --delete-tag

7.4.4 Modifying a Leader Value

The following set of options instructs **MARCBibUpdate.exe** to . . .

- Select all records with a value of *z* in position 17 of the Leader.
- Set the selected value to *u*.

Options:

--leader-string 17 z --replace-leader-string u

7.4.5 Replacing a Subfield

The following set of options instructs **MARCBibUpdate.exe** to . . .

- Select all records with an exact value of *Text format* in the 516 tag subfield $d.
- Change the subfield code for the selected subfield to *a*.

Options:

--tag-number 516 --subfield-code z --subfield-exactly "Text format" --change-subfield-code a

7.4.6 Using a Reverse Select

The following set of options instructs **MARCBibUpdate.exe** to . . .

- Select all records that do NOT have a 506 tag.
- Add a tag 590 subfield $a with a value of *No access restrictions*.

Options:

--tag-number 506 --reverse-select --add-tag 590 " $aNo access restrictions"

7.4.7 Printing Bibliographic IDs

The following set of options instructs **MARCBibUpdate.exe** to . . .

- Select all records that have an 039 with a second indicator of *9*.
- Print the bibliographic ID of each selected record to a file named **nonvtls_039**.

Options:

--tag-number 039 --second-indicator 9 --print-id

Note: If you use this set of options when the source data is a file of records, the program will print an ID number corresponding to the position of the record in the file rather than the bibliographic ID

7.4.8 Using Multiple Selection Blocks and a Single Action Block

The following set of options instructs **MARCBibUpdate.exe** to . . .

- Select all records with all of the following attributes:
  - An exact value of *Dumas* in the 039 tag subfield $z.
  - A substring value of 20021210 in the 039 tag subfield $y.
  - An exact value of *Text format* in the 516 tag subfield $d.

- Change the subfield code for subfield $d of the selected 516 tag to *a*.

Options:

--tag-number 039 --subfield-code z --subfield-exactly "Dumas" --tag-number 039 --subfield-code y --subfield-substring "20021210" --tag-number 516 --subfield-code z --subfield-exactly "Text format" --change-subfield-code a

About this example: This example is of one selection block that specifies three sets of criteria for selection:

- --tag-number 039 --subfield-code z --subfield-exactly "Dumas"
- --tag-number 039 --subfield-code y --subfield-substring "20021210"
- --tag-number 516 --subfield-code z --subfield-exactly "Text format"

To qualify for selection, a record must meet all three sets of the selection criteria. If a record is selected, the --change-subfield-code option specifies that the subfield code of the last selected subfield will be changed to *a*. Since the last selected subfield is the 516 subfield $d, this is the subfield code that will be changed.

7.4.9 Connecting Multiple Selection/Action Blocks

The following set of options instructs **MARCBibUpdate.exe** to . . .

- Select all records with all of the following attributes:
  - A value of *e* in position 6 of the Leader.
  - A 506 tag.
  - A 307 tag subfield $a with a value of exactly *10am-4pm*.
- Delete the 506 tag from selected records.
- Change the value of the 307 tag subfield $a to *7am-5pm*.

Options:

--leader-string 6 e --tag-number 506 --delete-tag --and-select --tag-number 307 --subfield-code a --subfield-exactly "10am-4pm" --replace-subfield "7am-5pm"

About this example: In this example, there are two selection blocks:

- --leader-string 6 --tag-number 506
- --tag-number 307 --subfield-code a --subfield-exactly "10am-4pm"

To qualify for selection, a record must meet the criteria specified in both selection blocks. The first selection block specifies two selection criteria. The action for that block is done on the last selection.

8. Working with Item Records

Virtua offers various utilities that let you modify specific aspects of item records. Additionally, Virtua provides facilities for deleting items in batch. This chapter provides details on the following topics, each of which deals with managing your item records:

8.1 Working with Item Statuses

Virtua offers the following utilities for working with item statuses:

- **addItemStatus.sh** - Adds item statuses to the specified records.
- **removeItemStatus.ksh** - Removes an item status from the specified records.
- **changeItemStatus.sh** - Changes an item status to a different item status in the specified records.
- **removeAddItemStatus.ksh** - Removes an item status from and adds an item status to the specified records.
- **ItemStatusMonitor.exe** - Finds items with a status that has expired and then moves it to the "Next Status" as defined in the Virtua Profiler.

8.1.1 Adding Item Statuses

To add statuses to a batch of item records,

1. Log in to your server as **dbadmin**.
2. At the prompt, type:

```
addItemStatus.sh [item-status file] [location]
```

Where . . .

- **[item-status file]** is the name of a file that specifies the item barcodes of the items to which you want add statuses and the status that you want to add. For each item that you want to modify, you must list them in the following format:

  ```
  [item barcode] [new status code]
  ```

  For example, if you want to . . .
  - Add the status code 4799 to an item with the barcode 5755387
  - Add the status code 4709 to an item with the barcode 7569389

  Type:

  ```
  5755387 4799
  7569389 4709
  ```

  Note: You cannot add a system-assigned status. For information about system-assigned statuses, see the *Virtua Profiler/Global Settings User's Guide*.

- **[location]** is the location *code* of the location for which you want to make these modifications.

3. Press Enter.

8.1.2 Removing Statuses

The **removeItemStatus.ksh** script allows you to remove a status from a group of item records. You can specify a file of the item barcodes or item IDs of the item IDs of items from which you want to remove a status. We describe each workflow in the sections below.

8.1.2.1 Removing Statuses By Item Barcode

To remove statuses from a batch of item records by providing a file of item barcodes,

1. Log in to your server as **dbadmin**.
2. At the prompt, type:

```
removeItemStatus.ksh [item-status file]
```

Where **[item-status file]** is the name of a file that specifies the item barcodes of the items from which you want to remove a status and the status that you want to remove. For each item that you want to modify, this file must use the following format:

```
[item barcode] [status code to remove]
```

For example, if you want to . . .

- Remove the status code 5703 from an item with the barcode 45283745
- Remove the status code 4709 from an item with the barcode 45645664

Type:

```
45283745 5703
45645664 4709
```

Note: You cannot remove a system-assigned status. For information about statuses, see the *Virtua Profiler/Global Settings User's Guide*.

3. Press Enter.

8.1.2.2 Removing Statuses By Item ID

To remove statuses from a batch of item records by providing a file of item IDs,

1. Log in to your server as **dbadmin**.
2. At the prompt, type:

```
removeItemStatus.ksh [item-status file] 1
```

Where **[item-status file]** is the name of a file that specifies the item IDs of the items from which you want to remove a status and the status that you want to remove. For each item that you want to modify, this file must use the following format:

```
[item ID] [status code to remove]
```

For example, if you want to . . .

- Remove the status code 5703 from an item with the item ID 45283745
- Remove the status code 4709 from an item with the item ID 45645664

Then type:

```
45283745 5703
45645664 4709
```

Note: You cannot remove a system-assigned status. For information about statuses, see the *Virtua Profiler/Global Settings User's Guide*.

3. Press Enter.

8.1.3 Changing Statuses

To change statuses in a batch of item records,

1. Log in to your server as **dbadmin**.
2. At the prompt, type:

```
changeItemStatus.sh [barcodes] [loc] [old status] [new status]
```

Where . . .

- **[barcodes]** is the name of a file that specifies the item barcodes of the items for which you want to change a status.
- **[loc]** is the location *code* of the location for which you want to make these modifications.
- **[old status]** is the code of the status that you want to change. This status will be removed from each item.
- **[new status]** is the code of the status that you want to add to records that have the status defined by the **[old status]** parameter. If an item record does not have the status defined by the **[old status]** parameter, the program will not add this status.

Note: You cannot change to or from a system-assigned status. For information about system-assigned statuses, see the *Virtua Profiler/Introduction Global Settings User's Guide*.

3. Press Enter.

8.1.4 Removing and Adding Statuses

To remove a status and add a status in a batch of item records,

1. Log in to your server as **dbadmin**.
2. At the prompt, type:

```
removeAddItemStatus.ksh [barcodes] [loc] [remove code] [add code]
```

Where . . .

- **[barcodes]** is the name of a file that specifies the item barcodes of the items for which you want to make modifications.
- **[loc]** is the location *code* of the location for which you want to make these modifications.
- **[remove code]** is the code of the status that you want to remove from each record.
- **[add code]** is the code of the status that you want to add to each record. This status will be added even if the item does not have the status that you want to remove.

Note: You cannot add or remove a system-assigned status. For information about system-assigned statuses, see the *Virtua Profiler/Global Settings User's Guide*.

3. Press Enter.

8.1.5 Updating Expired Item Statuses

In the Virtua Profiler, you can configure statuses to assign to item records. For each status, you can set an age threshold that determines how long it is assigned to an item. After the age threshold expires, the item is ready to move to the status designated as the "Next Status" for the item.

Note: For information about item statuses, see the *Virtua Profiler/Global Settings User's Guide*.

The program **ItemStatusMonitor.exe** runs at a library-defined interval to find any items with a status that has expired and then moves those items to the next status. You can run the executable with a user option, which will associate each new item status with the specified user and the user's log-in location.

To have your item statuses updated automatically, you need to run **ItemStatusMonitor.exe** as a background process. If you do not want the program to run as a background process, you can choose to have it check once for statuses that need to be updated and then exit.

Note: An item may have more than one status. Any status that has not yet expired will remain assigned to the item even if another status assigned to the item is removed.

8.1.5.1 Running ItemStatusMonitor.exe as a Background Process

To run ItemStatusMonitor.exe as a background process,

1. Log in to your server as **dbadmin**.
2. At the prompt, type:

```
(nohup ItemStatusMonitor.exe -i [interval] &)
-OR-
(nohup ItemStatusMonitor.exe --interval [interval] &)
```

Where **[interval]** is the interval (in minutes) at which you want the program to check for item statuses that need to be updated, i.e., moved to new statuses.

Tip: If you do not specify an interval, the program will check for item statuses every 60 minutes.

3. Press Enter.

The program checks for item statuses that need to be updated and moves them to the appropriate status. After it finishes updating statuses, the program will sleep for the interval you specified before starting again. It will continue this pattern until you kill the process.

8.1.5.2 Running ItemStatusMonitor.exe with the User Option

Running **ItemStatusMonitor.exe** with the user option associates all updated items statuses with the specified user and that user's log-in location.

To run ItemStatusMonitor.exe with the user option,

1. Log in to your server as **dbadmin**.
2. At the prompt, type:

   ```
   ItemStatusMonitor.exe -i [interval] -u [user]
   ```

   Where **[interval]** is the interval (in minutes) at which you want the program to check for item statuses that need to be updated and **[user]** is the username such as *staff* to be associated with the item status.

3. Press Enter.

Note: If you are running **ItemStatusMonitor.exe** in a consortium environment, the use of the user option is required and has different implications (for details, see the *Consortium Database User's Guide*).

8.1.5.3 Running ItemStatusMonitor.exe Once

To run ItemStatusMonitor.exe once,

1. Log in to your server as **dbadmin**.
2. At the prompt, type:

   ```
   ItemStatusMonitor.exe --interval 0
   ```

3. Press Enter.

The program checks for item statuses that need to be updated and moves them to the appropriate status. Once the program finishes updating item statuses, it exits until you choose to run it again.

8.2 Changing Item Locations

Virtua provides the following utilities for changing item locations:

- **ChangeLocation.sh** - Modifies the owning and shelving location of items that you specify in a file of Item-IDs OR barcodes.
- **ChangeShelfLocation.sh** - Modifies the shelving location of items that you specify in a file of Item-IDs or barcodes.

- **ChangeLocationByBarCode.sh** - Modifies the location of items that you specify in a file of barcodes.
- **ChangeLocationByItemId.sh** - Modifies the location of items that you specify in a file of Item-IDs. The Item-ID is the system-defined identifier for item records.
- **ChangeLocationByCallNumberRange.sh** - Modifies the shelving and owning location of items within a range of item-level call numbers.

8.2.1 Changing Owning and Shelving Locations

The script **ChangeLocation.sh** evaluates a list of item records and updates with the specified "new location" any item record that contains the specified "old location." This script updates both the owning and shelving location if both locations match the specified "old location." If the shelving location is different from the owning location, only the owning location will be changed when a match is found.

Note: If you want to change *only* the shelving location, see the section "Changing Shelving Locations" in this chapter. If the shelving location is modified, records will have to be reprocessed via **ReProcessBibs.ksh** (see the *Virtua System Management: OPAC User's Guide*) before the changes are reflected in your keyword search results.

This script can evaluate . . .

- Items that you specify in a file of Item-IDs or barcodes.
  -OR-
- All items in the database (if you do not specify an input file).

To change the owning location of item records,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type:

   ```
   ChangeLocation.sh [old loc] [new loc] [input type] < [input file]
   ```

   Where . . .

   - **[old loc]** is the location *code* of the existing location that you want to modify.

     Hint: Type **any** to update all records with the new location.

   - **[new loc]** is the location *code* of the location with which to replace the old location.

- **[input type]** specifies the type of input. Type:
  - **1** - To specify that the input file contains Item-IDs.
  - **2** - To specify that the input file contains item barcodes.
  - **3** - To specify NO input file, in which case ALL items in the database will be evaluated.

- **[input file]** is the name of the file of Item-IDs or barcodes.

  Note: If you set the input type to **3**, omit this argument and the preceding < sign.

3. Press Enter.

   Virtua evaluates the items and updates any old location with the new location. Any records that are modified will have their Last Modified Date (039 tag subfield $a) updated.

Note to consortia users: This script is designed to move items from one location to another location within a single institution. It should NOT be used to move items from a location at one institution to a location owned by a different institution. Doing so could cause inconsistencies in the database.

8.2.2 Changing Shelving Locations

The script **ChangeShelfLocation.sh** evaluates a list of item records and updates any item with the specified "old shelving location" with the "new shelving location." This script can evaluate . . .

- Items that you specify in a file of Item-IDs or barcodes.
  -OR-
- All items in the database (if you do not specify an input file).

Note: When you modify the shelving location by running this script, records will have to be reprocessed via **ReProcessBibs.ksh** (see the *Virtua System Management: OPAC User's Guide*) before the changes are reflected in your keyword search results.

To change the shelving location of item records,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type:

```
ChangeShelfLocation.sh [old loc] [new loc] [date] [input type] < [input file]
```

Where . . .

- **[old loc]** is the location code of the existing shelf location that you want to modify.

  Hint: Type **any** to update ALL records with the new shelving location.

- **[new loc]** is the location code of the shelving location with which to replace the old location.

- **[date]** is the new At Shelving Location Until date, which uses the input format YYYYMMDD.
- **[input type]** specifies the type of input. Type:
  - **1** - To specify that the input file contains Item-IDs.
  - **2** - To specify that the input file contains item barcodes.
  - **3** - To specify NO input file, in which case ALL items in the database will be evaluated.

- **[input file]** is the name of the file of Item-IDs or barcodes.

  Note: If you set the input type to **3**, omit this argument and the preceding < sign.

3. Press Enter.

   Virtua evaluates the items and updates the old shelving location with the new shelving location.

8.2.3 Changing Location by Barcode

To change the location of item records based on item barcodes,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type:

```
ChangeLocationByBarCode.sh [loc] < [barcode file]
```

Where . . .

- **[loc]** is the location *code* of the location to which you want to change the items.
- **[barcode file]** is a file that lists the barcodes of the item records for which the program will change the location.

3. Press Enter.

   The script changes the location code of the items specified in the file of barcodes to the location code that you specified in the command line.

Note to consortia users: This script is designed to move items from one location to another location within a single institution. It should NOT be used to move items from a location at one institution to a location owned by a different institution. Doing so could cause inconsistencies in the database.

8.2.4 Changing Location by Item ID

To change the location of item records based on the system-defined Item-ID,

1. Log in to your server as **dbadmin**.
2. At the prompt, type:

   ```
   ChangeLocationByItemId.sh [loc] < [ID file]
   ```

   Where . . .

   - **[loc]** is the location *code* of the location to which you want to change the items.
   - **[ID file]** is a file that lists the system-defined IDs of the item records for which the program will change the location.

3. Press Enter.

   The script changes the location code of the items specified in the file of IDs to the location code that you specified in the command line.

Note to consortia users: This script is designed to move items from one location to another location within a single institution. It should NOT be used to move items from a location at one institution to a location owned by a different institution. Doing so could cause inconsistencies in the database.

8.2.5 Changing Location by a Range of Item-level Call Numbers

To change the location and/or shelving location of item records based on a range of item-level call numbers,

1. Log in to your server as **dbadmin**.
2. At the prompt, type:

   ```
   ChangeLocationByCallNumberRange.sh
   ```

3. Respond to the interactive prompts that appear on the screen.

4. Press Enter.

The location code(s) of the items are changed to the location code(s) that you specified in the script.

8.3 Changing a Location Code

When you move items from one location to another, all the associated database tables have to be updated. Virtua provides a script that updates ALL tables that store the location code or location name, replacing the old location code with the new location code. The script also updates the data in the holdings record, tag 852 subfield $b.

Use the script **ChangeLocationId.sh** to change one location code to a new location code. The script prompts for an old location code and a new location code.

Note:

- If the script is being used to change a sublocation to a main location or main location to a sublocation, you MUST create the sublocation or main location first via the Virtua Profiler and set all parameter settings. The script can then be used to move all associated data from the one existing location to another.
- If the script is being used to create a brand new location, the value for **NewLocation** must be of the same location type as **OldLocation**; e.g., if **OldLocation** is a main location, then **NewLocation** must also be a main location, with a location code that ends in 0000.
- If you change a main location ID, then you MUST also change the location IDs of all of the sublocations of that main location, so that all the sublocations have a location code that begins with the same digits as the new main location code.

To run ChangeLocationId.sh,

1. Log in to your server as **dbadmin.**
2. At the prompt, type:

   ```
   ChangeLocationId.sh [OldLocation] [NewLocation]
   ```

   Where...

   - **[OldLocation]** is the old location code that you want to change to a new location code.
   - **[NewLocation]** is the new location code.

3. Press Enter.

Virtua changes the location ID from the old location code to the new location code in ALL tables that use it. To complete the process, you must now reindex Chamo, run **KeywordIndex.exe**, and run **PopulateMissingLocationFilters.sh**.

Note: The script will display a warning message if the new location code you enter already exists. This is to prevent staff users from accidentally re-using an existing location code for a brand new location.

8.4 Changing Item Classes

Virtua provides the following utilities for modifying item classes:

- **ChangeItemClass.sh** - Modifies the item class of records that you specify in a file of Item-IDs or barcodes. The item classes you choose must be "Regular" item classes.
- **ChangeReserveItemClass.sh** - Modifies the reserve item class of records that you specify in a file of Item-IDs or barcodes.

8.4.1 Modifying Item Classes

The script **ChangeItemClass.sh** evaluates a list of item records and updates any item with the specified "old item class" with the "new item class." This script can evaluate . . .

- Items that you specify in a file of Item-IDs or barcodes.

-OR-

- All items in the database (if you do not specify an input file).

To change the item class of item records,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type:

   ```
   ChangeItemClass.sh [old item class] [new item class] [input type]
   < [input file]
   ```

   Where . . .

   - **[old item class]** is the item class code of the existing item class that you want to modify. This MUST be a Regular item class type; using a Bookable or Reserve item class will cause an error.

     Hint: Type **any** to update all records with the new item class.

   - **[new item class]** is the item class code of the item class with which to replace the **[old item class]**. This MUST be a regular item class; using a Bookable or Reserve item class will cause an error.

   - **[input type]** specifies the type of input. Type:
     - **1** - To specify that the input file contains Item-IDs.
     - **2** - To specify that the input file contains item barcodes.
     - **3** - To specify NO input file, in which case ALL items in the database will be evaluated to see if they have the specified **[old item class]**.

   - **[input file]** is the name of the file of Item-IDs or barcodes.

     Note: If you set the input type to **3**, omit this argument and the preceding **<** sign.

3. Press Enter.

   Virtua evaluates the items and updates any old item class it finds with the new item class.

8.4.2 Modifying Reserve Item Classes

The script **ChangeReserveItemClass.sh** evaluates a list of item records and updates any item with the specified "old reserve item class" with the "new reserve item class." This script can evaluate . . .

- Items that you specify in a file of Item-IDs or barcodes.
  -OR-
- All items in the database (if you do not specify an input file).

To change the reserve item class of item records,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type:

   ```
   ChangeReserveItemClass.sh [old reserve item class] [new reserve item class] [input type] < [input file]
   ```

   Where . . .

   - **[old reserve item class]** is the item class code of the existing reserve item class that you want to update.

     Hint: Type **any** to update all records with the new item class.

   - **[new reserve item class]** is the item class code of the reserve item class with which to replace the **[old reserve item class]**.

   - **[input type]** specifies the type of input. Type:
     - **1** - To specify that the input file contains Item-IDs.
     - **2** - To specify that the input file contains item barcodes.
     - **3** - To specify NO input file, in which case ALL items in the database will be evaluated to see if they have the specified **[old reserve item class]**.

   - **[input file]** is the name of the file of Item-IDs or barcodes.

     Note: If you set the input type to **3**, omit this argument and the preceding < sign.

3. Press Enter.

Virtua evaluates the items and updates any old reserve item class it finds with the new reserve item class.

Note that the **ChangeReserveItemClass.sh** script updates item records ONLY. If you wish to . . .

- Update the list of reserve item classes that are available to be selected when adding an item to a reserve list, you must modify an item class's Class Type via the Item Class Definitions parameter. See the *Virtua Profiler/Circulation Parameters User's Guide* for more information.
- Update the default reserve item class that is designated for a particular location, you must modify the Default Reserve Item Class option in the Location Names parameter. See the *Virtua Profiler/Global Settings User's Guide* for more information.

8.5 Modifying the Item Price

The script **ChangeItemPrice.sh** evaluates a set of item records and updates a specified item price with a different item price. This script can evaluate . . .

- Items that you specify in a file of Item-IDs or barcodes.
  -OR-
- All items in the database (if you do not specify an input file).

To change the item price of a specified set of item records,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type:

   ```
   ChangeItemPrice.sh [old item price] [new item price] [input type]
   < [input file]
   ```

   Where . . .

   - **[old item price]** is the existing item price that you want to update.

     Hint: Type **any** to update all records with the new item price.

   - **[new item class]** is the new price with which to replace the **[old item price]**.

   - **[input type]** specifies the type of input. Type:
     - **1** - To specify that the input file contains Item-IDs.
     - **2** - To specify that the input file contains item barcodes.
     - **3** - To specify NO input file, in which case ALL items in the database will be evaluated to see if they have the specified **[old item price]**.

   - **[input file]** is the name of the file of Item-IDs or barcodes.

> **Note:** If you set the input type to **3**, omit this argument and the preceding < sign.

3. Press Enter.

   Virtua evaluates the specified set of items and updates any "old item price" it finds with the "new item price."

8.6 Modifying Item-level Call Numbers

Virtua offers the following scripts for modifying item-level call numbers:

- **AddItemCallNumberPrefix.sh** - Adds a string to the beginning of item-level call numbers.
- **ReplaceSubstringInItemCallNumber.sh** - Replaces a substring in item-level call numbers with another string.
- **RemoveBlankItemCallNum.sh** - Removes item-level call numbers that consist only of space characters.

These scripts can be useful in standardizing item-level call numbers in your database.

8.6.1 Adding an Item-level Call Number Prefix

The script **AddItemCallNumberPrefix.sh** lets you add a string to the beginning of existing item-level call numbers. These items are specified by a file of item IDs that are uploaded to your server before you run the script.

To add a prefix to item-level call numbers,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type:

```
AddItemCallNumberPrefix.sh [userid] [prefix] < [filename]
```

Where . . .

- **[userid]** is the user ID to be used as the operator for processing,
- **[prefix]** is the string to be prepended to the existing call numbers, and
- **[filename]** is the name of a file containing the item IDs of the items to have the prefix attached to the beginning of the items' call numbers.

The script modifies the item-level call numbers of the specified items. The operation will be recorded in the Operation_Log table in the Virtua database.

Note: This script can NOT be used to modify item-level call numbers that are inherited from the bibliographic record.

8.6.2 Replacing Characters in Item-level Call Numbers

The script **ReplaceSubstringInItemCallNumber.sh** lets you identify a string in existing item-level call numbers, and specify a new string to replace that string. The items are specified by a file of item IDs that are uploaded to your server before you run the script.

To replace a substring in a list of item-level call numbers,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type:

```
ReplaceSubstringInItemCallNumber.sh [userid] [string]
[replacement] < [filename]
```

Where . . .

- **[userid]** is the user ID to be used as the operator for processing,
- **[string]** is the string of characters to be removed from the call numbers,
- **[replacement]** is the string that will replace the removed string, and
- **[filename]** is the name of a file containing the item IDs of the items to have a substring replaced.

Note: This script can NOT be used to modify item-level call numbers that are inherited from the bibliographic record.

8.6.3 Removing Blank Item-level Call Numbers

The script **RemoveBlankItemCallNum.sh** removes item-level call numbers that consist only of space characters.

To remove item-level call numbers,

1. Log in to your server as the **dbadmin** user.

2. At the prompt, type:

   RemoveBlankItemCallNum.sh

Item-level call numbers with only space characters will be removed. This means that pre-2013.1 versions of **CallIndexForm.sh** will be able to successfully index call numbers. See the *System Management: OPAC User's Guide* for information on **CallIndexForm.sh**.

8.7 Modifying Circulation Rules

Virtua provides the following utilities for modifying circulation rules:

- **ChangeLoanPeriod.sh** - Modifies the loan rules of the items that you specify in a file of Item-IDs or barcodes.
- **ChangeAllowRequest.sh** - Modifies the Allow Request setting of items that you specify in a file of Item-IDs or barcodes.

8.7.1 Modifying the Loan Period

The script **ChangeLoanPeriod.sh** lets you modify the loan rules for a specified set of item records. This script can evaluate . . .

- Items that you specify in a file of Item-IDs or barcodes.
  -OR-
- All items in the database (if you do not specify an input file).

To modify the loan rules in a specified set of item records,

3. Log in to your server as the **dbadmin** user.
4. At the prompt, type:

   ChangeLoanPeriod.sh [new loan period flag] [old loan period] [new loan period] [input type] < [input file]

   Where . . .

   - **[new loan period flag]** specifies the new loan period type that you want to assign to items. Type:
     - 0 - To set the circulation rules to Use Matrix.
     - 1 - To set the circulation rules to Use Loan Period.

- **[old loan period]** is the old loan period (in days) that you want to update. Records will be modified only if they have the loan period that you specify here.

  Hint: Type **`any`** to update all records with the new loan period flag and/or loan period.

- **[new loan period]** is the new loan period with which to replace the **[old loan period]**. This value will be modified only in records where the loan period flag is set to Use Loan Period.

  Hint: Type **`same`** if you want to modify only the loan period flag and not change the loan period itself.

- **[input type]** specifies the type of input. Type:
  - **`1`** - To specify that the input file contains Item-IDs.
  - **`2`** - To specify that the input file contains item barcodes.
  - **`3`** - To specify NO input file, in which case ALL items in the database will be evaluated to see if they qualify for a loan period change based on the options that you set.

- **[input file]** is the name of the file of Item-IDs or barcodes.

  Note: If you set the input type to **3**, omit this argument and the preceding < sign.

5. Press Enter.

   Virtua evaluates the specified set of items and updates the loan period for all qualifying items.

8.7.2 Modifying the Allow Request Setting

The script **ChangeAllowRequest.sh** evaluates a set of item records and updates a specified item price with a different item price. This script can evaluate . . .

- Items that you specify in a file of Item-IDs or barcodes.
  -OR-
- All items in the database (if you do not specify an input file).

To change the item price of a specified set of item records,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type:

   ```
   ChangeAllowRequest.sh [request setting] [input type] < [input file]
   ```

   Where . . .

   - **[request setting]** is the Allow Request setting that you want to assign to the specified items. Type:
     - **0** - NOT to allow requests on the specified items.
     - **1** - To allow requests on the specified items.
   - **[input type]** specifies the type of input. Type:
     - **1** - To specify that the input file contains Item-IDs.
     - **2** - To specify that the input file contains item barcodes.
     - **3** - To specify NO input file, in which case the new request setting will be applied to ALL items in the database.
   - **[input file]** is the name of the file of Item-IDs or barcodes.

     Note: If you set the input type to **3**, omit this argument and the preceding < sign.

3. Press Enter.

   Virtua evaluates the specified set of items and updates any all items with the new Allow Request setting.

8.8 Deleting Item Records from the Database

Virtua offers the following scripts for deleting item records from the database:

- **BatchDeleteItems.exe** - Deletes from the database item records specified in a file of Item-IDs.
- **DeleteItemsByBarcode.sh** - Deletes from the database item records specified in a file of item barcodes.
- **DeleteItemsByLocation.sh** - Deletes from the database items associated with the specified location code.

- **DeleteItemsByDueDate.sh** - Deletes from the database all items that were due (and still checked-out) *before* the specified date. It allows you to enter a location ID as an optional parameter to limit the items deleted to a specific location; consortium customers must enter both an institution code and a location ID.
- **DeleteItemsByStatus.sh** - Deletes from the database all items that carry the specified status. It allows you to enter a location ID as an optional parameter to limit the items deleted to a specific location; consortium customers must enter both an institution code and a location ID.

Note: With the exception of **DeleteItemsByDueDate.sh** and **DeleteItemsByStatus.sh**, when you delete item records using these scripts, Virtua performs the same checks as it does when you delete an item record via the Virtua client. In this way, these scripts will delete only the items that qualify for deletion according to standard item deletion rules. For example, these scripts will NOT delete items that have current circulate activity or special statuses. For additional information on how items qualify for deletion, see the *Virtua Cataloging User's Guide*.

8.8.1 Deleting Items by Item-ID

Note: In order for **BatchDeleteItems.exe** to function, in the Virtua Profiler the Root User must have selected in the Log-in Locations setting the location of any item listed in the input file. If the Root User does not have the Log-in Locations selected for a specific location that an item belongs to, the following error will appear and the item will not be deleted:

```
>>>>> Could not delete item id: XXXXX
Error - User is not associated with the item's location.
```

To delete a specific set of item records,

1. Log in to your server as the **dbadmin** user.

2. At the prompt, type:

   ```
   BatchDeleteItems.exe [command option] < [input file]
   ```

   Where the command option **-u** [--check-unserviceable] can be used to specify that the program notify (and NOT update) when the last item is "unserviceable." Default is *false*.

   -AND-

Where **[input file]** is the name of the file that contains the list of Item-IDs.

3. Press Enter.

   Virtua deletes from the database the item records associated with the Item-IDs in the specified file.

8.8.2 Deleting Items by Barcode

To delete specific set of item records,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type:

   ```
   DeleteItemsByBarcode.sh < [input file]
   ```

   Where **[input file]** is the name of the file that contains the list of item barcodes.

3. Press Enter.

   Virtua deletes the database the item records associated with the barcodes in the specified file.

8.8.3 Deleting Items by Location

To delete item records associated with a specified location,

- Log in to your server as the **dbadmin** user.
- At the prompt, type:

  ```
  DeleteItemsByLocation.sh [location code]
  ```

  Where **[location code]** is the location code of the location for which you want to delete items.

- Press Enter.

  Virtua deletes from the database all the item records associated with the specified location.

8.8.4 Deleting Items by Due Date

To delete item records by due date,

1. Log in to your server as the **dbadmin** user.

2. At the prompt, type:

   ```
   DeleteItemsByDueDate.sh [Due_Date (YYYYMMDD)] [Location_Id]
   [Institution_Symbol]
   ```

   Where **[Due_Date (YYYYMMDD)]** is a required parameter and **YYYMMDD** is the format of the due date. **[Location_Id]** is an optional parameter to restrict the item records that are deleted. **Note:** For consortium databases, both the Location ID and the Institution Symbol are *required* parameters.

3. Press Enter.

   All item records that have a due date *earlier* than the input date will have their circulation transactions and/or status deleted and then the item record itself will be deleted.
   - If a location ID is provided, only an item record whose owning location matches the entered location AND has a qualifying due date will be deleted.
   - If a main location is entered, all of the sublocations will be included during item selection.
   - If the items deleted by this script leave any bibliographic records without any items attached, those bibliographic records will be deleted and their control numbers written to a file.

   The script will output the following files:

   - **itemIds.deleted** - A list of the Item-IDs of deleted items.
   - **bibIds.deleted** - A list of the Bib-IDs of deleted bibliographic records.
   - **controlNumbers.deleted** - A list of Bib-IDs and control numbers of deleted bibliographic records.

8.8.5 Deleting Items by Status

To delete item records by status,

1. Log in to your server as the **dbadmin** user.

2. At the prompt, type:

   ```
   DeleteItemsByStatus.sh [status_code] [Location_Id]
   [Institution_Symbol]
   ```

   Where **[status_code]** is a required parameter and **[Location_Id]** is an optional parameter to restrict the item records that are deleted. **Note:** For consortium databases, both the Location ID and the Institution Symbol are *required* parameters.

3. Press Enter.

   All item records that carry the specified status will have their circulation transactions and/or status deleted and then the item record itself will be deleted.
   - If a location ID is provided, only an item record whose owning location matches the entered location AND has a qualifying status will be deleted.
   - If a main location is entered, all of the sublocations will be included during item selection.
   - If the items deleted by this script leave any bibliographic records without any items attached, those bibliographic records will be deleted and their control numbers written to a file.

   The script will output the following files:

   - **itemIds.deleted** - A list of the Item-IDs of deleted items.
   - **bibIds.deleted** - A list of the Bib-IDs of deleted bibliographic records.
   - **controlNumbers.deleted** - A list of Bib-IDs and control numbers of deleted bibliographic records.

9. Working with Serials Patterns

In this chapter we discuss two scripts that are used in Serials Control. One is for extracting serials patterns from holdings records, and the other is for loading the contents of a pattern file to the database for use with the Serials Pattern Editor.

9.1 Extracting Serials Patterns

The script **List853Holdings.sh** generates a list of the prediction patterns from the holdings records stored in your database. In a holdings record, prediction patterns are stored in the 853 tag. You can review the prediction patterns generated by this script to find invalid prediction patterns.

To generate a list of prediction patterns from the holdings records in your database,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type: **`List853Holdings.sh`**
3. Press Enter.

The script scans the holdings records in your database and creates two files:

- **holdingsIdsWith853.dat** - Lists the holdings ID (001 tag) of each record that has one or more 853 tags.
- **holdingsIdsWith853.print** - Lists the holdings ID (001 tag) and the 853 tags of the holdings record in your database.

You can use the information in the file **holdingsIdsWith853.print** to determine if the prediction patterns conform to the MARC 21 Format for Holdings Data standard. For information about this standard, go to *http://www.loc.gov/marc/holdings/echdhome.html*.

9.2 Loading Serials Patterns

.

To load the patterns in pattern.txt **to the database,**

1. Log in to your server as the **dbadmin** user.

2. At the prompt, type: **`LoadPatterns.ksh`**

3. Press Enter.

Virtua loads the file of patterns into the Saved_Serials_Patterns table in the database. Using the Serials Pattern Editor in the Virtua client, you can retrieve these patterns for insertion into holdings records

10. Working with Conversion Programs

This chapter discusses the following topics:

10.1 Converting UNIMARC Authorities to MARC 21 Format

The program **ToMARC21.exe** converts files of UNIMARC authority records to MARC 21 format. Using this program, you can prepare a file of UNIMARC authority records to be loaded to the Virtua database.

To convert UNIMARC authorities to MARC 21 format,

1. Log in to your server as the **dbadmin** user.

2. At the prompt, type: **`ToMARC21.exe < [input file] > [output file]`**

   Where . . .
   - **[input file]** is the name of the file of UNIMARC authorities.
   - **[output file]** is the name of the file to which you want to write the converted records.

3. Press Enter.

The program writes to the output file the converted records.

Tip: You can use the following command to pipe records converted by **ToMARC21.exe** directly to **vload.exe**:

```
ToMARC21.exe < authorities.rec | vload.exe &
```

When you pipe the converted records directly to **vload.exe**, they will be loaded to the database as they are processed by **ToMARC21.exe**.

10.2 Converting MARCXML Records to MARC 2709

The executable **XMLToMARC.exe** converts a file of MARCXML records to MARC 2709 records. After the records are converted, they can be loaded into Virtua using **vload.exe**.

To run **XMLToMARC.exe**,

1. Log in to your server as **dbadmin**.
2. At the database prompt, type:

   ```
   XMLToMARC.exe < [input filename] > [output filename]
   ```

   Where **[input filename]** is the name of a file of MARC XML records in your executable directory, and **`[output filename]`** is the name you'd like to give to the output file of MARC records.
3. Press Enter.

The program runs and converts the MARCXML files to MARC 2709 files with the filename you specified. The program will display an error if it encounters any problems with the MARCXML files.

10.3 Converting MARC 2709 Records to MARCXML

The executable **2709ToXML.exe** converts a file of MARC 2709 records to MARCXML records.

To run **2709ToXML.exe**,

1. Log in to your server as **dbadmin**.

2. At the database prompt, type:

   `2709ToXML.exe < [input filename] > [output filename]`

   Where **[input filename]** is the name of a file of MARC 2709 records in your executable directory, and **`[output filename]`** is the name you'd like to give to the output file of MARCXML records.

3. Press Enter.

   The program runs and converts the MARC 2709 files to MARCXML files with the filename you specified. The program will display an error if it encounters any problems with the MARC 2709 files.

10.4 Converting MARC Records to a Different Character Set

The executable **ConvertMARCFileCharacterSet.exe** can be used to convert a file of MARC records from a specified character to the UTF-8 character set or from the UTF-8 character set to another character set. This program is especially useful for converting MARC records from a specified character set to UTF-8, the character set of a Virtua database, for special processing.

To run ConvertMARCFileCharacterSet.exe,

1. Log in to your server as **dbadmin**.

2. At the prompt, type:

`ConvertMARCFileCharacterSet.exe –s[source character set code] –d [destination character set code] –i[optional input filename] > [output filename]`

Where **[character set code]** is one of the following codes:

- **2** - UTF-8
- **10** - MARC-8

- **11** - ANSI Z39.47 (ANSI-8)
- **12** - EUROPA-3
- **13** - Windows Latin1
- **14** - PC-8
- **15** - ALA
- **16** - Windows Arabic
- **17** - Windows Hebrew
- **18** - Windows Cyrillic
- **19** - Windows Latin2
- **20** - ISO 6937-2
- **21** - CP-850 (Microsoft Code Page)
- **22** - ISO 6937-2 + Arabic
- **23** - ISO 6937-2 + Greek
- **24** - Big5 (Tamkang version)
- **25** - ANSI-8 + Hebrew
- **26** - UTF-8 character set where some separate diacritics need to be combined
- **28** - ANSI Z39.47 (ANSI-8) - Swiss version
- **30** - GBK (encoding of CJK characters)
- **31** - TIS620 Classic Thai
- **32** - ISO 5426 (International Serials Data System interchange)
- **33** - CCCII (Chinese Character Code for Information Interchange)
- **34** - GB 18030 (Chinese National Standard GB 18030-2000)
- **35** - Big5-HKSCS (Hong Kong Special Character Set)

If you do not specify an input filename, the program will use **`stdin`**.

10.5 Converting ISO-2709, MARCXML, and MODSXML Records

The program **RecordConverter.exe** converts a file of records that are in one of the following three formats—ISO-2709, MARCXML or MODSXML—to a file of records in either ISO-2709 format or MARCXML format.

If you plan to use the program to convert MODSXML records, first run the script **scripts/LoadRecordConverterFiles.sh** to place and store a style sheet on the server. You need to run the script only once before using the RecordConverter.exe program for the first time.

To convert a file of records from one format to another,

1. Log in to your server as the **dbadmin** user.

2. At the prompt, type:

```
RecordConverter.exe -aX -bY < [name of input file to convert] > [name
of converted output file]
```

Where . . .
-a is the input record format
-b is the output record format

X is **0, 1 or 2**
and
Y is **0** or **1**

And where…
0=ISO-2709
1=MARCXML
2-MODSXML

3. Press Enter.

The program writes the converted records to the output file.

10.6 Loading RDA Translations and Converting AACR2 to RDA in MARC Records

Note: For details of the actual mapping from AACR2 cataloging elements to RDA cataloging elements in MARC records, see Appendix C.

Two scripts are available for working with Resource Description & Access (RDA) translations in the database: **ConvertMARCRecordsToRDA.sh** and **ImportRDATranslations.sh**.

- **ConvertMARCRecordsToRDA.sh** loads RDA translations into the database and then calls the executable **ConvertMARCRecordsToRDA.exe** with the user supplied parameters to complete the conversion of the user supplied file of MARC records.

- **ImportRDATranslations.sh** is a Data Pump import script that is called by **ConvertMARCRecordsToRDA.sh**. It can be run as a standalone script to load either the bibliographic or authority RDA translations into the database. It will load one of the following two Oracle Data Pump export files as determined by user input when running the script:

  - **RDAAuthTranslations.dmp** - Data Pump export file of Authority RDA translations.
  - **RDABibTranslations.dmp** - Data Pump export file of Bibliographic RDA translations.

10.6.1 Running ConvertMARCRecordsToRDA.sh

To load RDA translations and convert AACR2 elements to RDA elements in MARC records,

1. Log in to your server as the **dbadmin** user.

2. At the prompt, type:

```
ConvertMARCRecordsToRDA.sh <inputfile> [Auth|Bib]
[remove_subfield_245_h 0|1] [add_33x_tag 0|1]
```

Where the optional input parameters are as follows:

[Auth|Bib] - Determines whether only Authority records OR Bibliographic records are to be converted during a single run of this script. Type **`Auth`** or **`Bib`**. The default is Bib.

[remove_subfield_245_h 0|1] - Determines whether the subfield 245 $h is removed from the record during the conversion. Type **`0`** or **`1`**. The default is 0 (will not remove subfield).

[add_33x_tag 0|1] - Determines whether tags 336, 337, and 338 are added to the record based on the data in 245 $h. Type **`0`** or **`1`**. The default is 1 (will add tags).

3. Press Enter.

The script will call the helper file **ImportRDATranslations.sh** to load the appropriate RDA translations from the $EXE_DIR and then will call the executable **ConvertMARCRecordsToRDA.exe** to do the conversion of the MARC records.

The script will store the log file with the results from the Data Pump job in the current directory.

Note: If a bibliographic record already contains **$e rda** or **$e rdax** in the 040 tag, the **ConvertMARCRecordsToRDA.exe** program will NOT modify the record.

Examples:

- To convert the file of *bibliographic* records while NOT removing the subfield 245 $h data and NOT adding 33X tags, type:
  `$EXE_DIR/ConvertMARCRecordsToRDA.sh`

- To convert the file of *authority* records while removing the subfield 245 $h data and NOT adding 33X tags, type:
  $**`EXE_DIR/ConvertMARCRecordsToRDA.sh Auth 1 0`**

10.6.2 Running ImportRDATranslations.sh in Standalone Mode

ImportRDATranslations.sh uses the Oracle Data Pump import utility to import the RDA translations for either Bibliographic records (**RDABibTranslations.dmp**) or Authority records (**RDAAuthTranslations.dmp**) into the RDA tables. Only bibliographic OR authority translations can be stored in the database at a time. You cannot load both.

Note: ImportRDATranslations.sh must be run from the $EXE_DIR directory, OR the .dmp files must be copied into the directory from which this script is executed.

To load RDA translations for MARC records by running ImportRDATranslations.sh as a standalone program,

1. Log in to your server as the **dbadmin** user.

2. At the prompt, type: **`ImportRDATranslations.sh [Auth|Bib]`**

   Where **[Auth|Bib]** determines whether only Authority records OR Bibliographic records are to be loaded. Type **`Auth`** or **`Bib`**.

3. Press Enter.

   The data will be loaded.

10.6.3 Running ConvertMARCRecordsToRDA.exe in Standalone Mode

By automatically calling **ConvertMARCRecordsToRDA.exe**, the script **ConvertMARCRecordsToRDA.sh** offers an all-in-one program for loading RDA translation files into the database and converting a file of MARC bibliographic or authority records from descriptive cataloging form (AACR2) to resource description and access cataloging form (RDA). However, if you want to take advantage of several more options (-c, -f, and –g) offered by **ConvertMARCRecordsToRDA.exe**, you will want to execute it directly from the command line.

Note: Before running **ConvertMARCRecordsToRDA.exe**, you need to run **ImportRDATRanslations,sh** first for either bibliographic or authority records.

To run ConvertMARCRecordsToRDA.exe,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type: **`ConvertMARCRecordsToRDA.exe [options] < AACRInputfileName > RDAOutputFileName`**

   Where **[options]** consist of any of the following:

   -? [--help] - Print usage statement.

   -a [--is-authority-record] - Input record is an authority record. *Default*: false.

   -h [--remove-GMD] - Remove the GMD 245 $h. *Default*: Do NOT remove 245 $h.

   -3 [--do-not-add-33x-tags] - Do NOT add 336, 337, 338 tags based on the GMD 245 $h. *Default*: Add the 33x tags.

   -c [--write-only-changed] - Write only records that are changed to the output. *Default*: Write all records.

   -f [--GMDTo33x-filename] arg - Filename of GMDTo33x maps: $EXE_DIR/migrate/GMDTo33xMap.txt (*Default*)

   -g [--which338-filename] arg - Filename of which338 maps: $EXE_DIR/migrate/which338Map.txt (*Default*)

3. Press Enter.

The MARC records will be converted per the specified options.

Note: If a bibliographic record already contains **$e rda** or **$e rdax** in the 040 tag, the **ConvertMARCRecordsToRDA.exe** program will NOT modify the record.

11. Enabling Multiple Subject Headings

To enable multiple subject headings, you must configure settings in the Profiler AND run several scripts in a specified order against your sever. The steps for enabling multiple subject headings are outlined in the section below. The steps in the following section provide only an overview of the required Profiler settings. You will find detailed information about these settings in the *Virtua Profiler/Cataloging Parameters User's Guide.*

Important: Multiple subject headings functionality is available to your library for an additional charge and requires that you complete several steps in a particular order. Before you run any scripts associated with enabling multiple subject headings, contact Innovative Customer Services. Some of the functions associated with enabling multiple subject headings can damage your database if proper preparations have not been made.

Keep in mind that you CANNOT disable multiple subject heading functionality once it is enabled.

This chapter covers the following topics:

11.1 Overview of Steps for Enabling Multiple Subject Headings

1. In the Profiler, define subject headings to be indexed.

2. On the server, run **CreateSubjectThesauri_1.sh**, as described in the section "Identifying Problems with Authority Records" of this guide, against the database. This script will identify any problems with the specified authority records.

3. On the server, run **UpdateAuthoritySubj.exe**, as described in the section "Correcting Invalid Subject Indicators in Authority Records" of this guide, to fix any problem authority records identified.

4. In the Profiler, enable Multiple Subject Headings.

5. On the server, run **CreateSubjectThesauri_2.sh**, as described in section "Creating Indexes and Duplicate Subject Headings" of this guide, against the database.
   This script…
   - Uses the file of bib-ids generated in **CreateSubjectThesauri_1.sh** to create duplicate subject headings for subject headings in more than one thesaurus.
   - Creates indexes for each thesaurus.

6. In the Profiler, set User Permissions to view the subject heading(s) in the OPAC as desired.

11.2 Identifying Problems with Authority Records

Important: Before you take this step, make sure you have completed the first step described in the section "Overview of Steps for Enabling Multiple Subject Headings" of this guide:
1. Define subject headings to be indexed

After you have defined subject headings to be indexed, you will need to run the script **CreateSubjectThesauri_1.sh** against the database to identify any problems with authority records.

To identify any problems with authority records,

1. Log in to your server as the **dbadmin** user.

2. At the prompt, type:

   ```
   CreateSubjectThesauri_1.sh
   ```

3. Press Enter.

The program writes to the output file any problems with authority records.

If the program output indicates that you have authority records with invalid subject indicators, you will need to run **UpdateAuthoritySubj.exe**, which is described in the following section.

Note: If you have bibliographic records in the database with a second indicator value representing an index that has not been enabled in the Profiler, **CreateSubjectThesauri_1.sh** will include in the output a list of associated Bib IDs.

If you have selected all indexes you need, this information should not be regarded as an error message. However, be aware that these records may not appear in a general subject search unless they contain a 6xx tag with an indexed subject heading.

11.3 Correcting Invalid Subject Indicators in Authority Records

Important: Before you take this step, make sure you have completed the first two steps described in the section "Overview of Steps for Enabling Multiple Subject Headings" of this guide.

1. Define subject headings to be indexed
2. Run **CreateSubjectThesauri_1.sh**

If the output of **CreateSubjectThesauri_1.sh** indicates that you have Authority records with invalid subject indicators, you will need to run the program **UpdateAuthoritySubj.exe**.

Multiple Subject Headings uses the fixed field 008 tag position 011 of the authority record to determine to which thesaurus the subject authority belongs. The purpose of the program **UpdateAuthoritySubj.exe** is to read a file of authority IDs and update any invalid 008 tag position 011 of the specified authority records.

The basic format for running **UpdateAuthoritySubj.exe** is:

UpdateAuthoritySubj.exe [command options] < [input file] > [output file]

Where . . .

- **[command options]** are the optional parameters that you can use to specify attributes such as default values and indicators to use. For a description of the available command-line options, see the section "Command-line Options for UpdateAuthoritySubj.exe" in this chapter.

- **[input file]** is the name of the file which lists, line-by-line, the Authority IDs of the authority records that you want to update.
- **[output file]** is the file to which you want **UpdateAuthoritySubj.exe** to extract the records.

11.3.1 Command-line Options for UpdateAuthoritySubj.exe

This section explains each of the options that are available for use with **UpdateAuthoritySubj.exe**. You can specify these options on the command line as described in the previous section. We recommend that you read the contents of this section before running this program.

For each command-line option, the following information is provided:

- **Description** - A brief explanation of what the option does (i.e., how it affects the update process).
- **Format** - The format that you must use to call the option at the command line.
- **Example** - An example of the syntax that you will use to call the option on the command line.

Note: If no options are set, the system will use the following default, which tells the system to use the first associated bibliographic record to set authority record values, and if no bibliographic records are attached, to output Authority IDs only: **–b0**

11.3.1.1 Only Report the Invalid Authority Records (-R)

Description: Specifies that no updates will occur. Authority IDs of invalid authority records will be output to the specified file.

Format: `-R`

Example: If you want to output Authority IDs only, without making updates, type:

```
UpdateAuthoritySubj.exe -R < input file > output file
```

The program will check the authority records specified in the **input file** and output the Authority IDs of any invalid authority records to the specified **output file**.

11.3.1.2 Do Not Check Bibliographic Records and Use Default Values to Update Records (-D)

Description: Specifies that the system should not check for associated bibliographic records and should instead use default values to update authority records.

Format: `-D`

Example: If you do not want the system to check for associated bibliographic records, type:

```
UpdateAuthoritySubj.exe -D < input file > output file
```

The program will ignore bibliographic records and use default values to update all records specified in the **`input file`**.

11.3.1.3 Define Default Character Value for 008 Tag position 011 (-i)

Description: Specifies which character should be used as the default value for the 008 tag position 011.

Format: `-i`

Example: If you want the default character value to be z, type:

```
UpdateAuthoritySubj.exe -iz < input file > output file
```

The program will use z as the default character value for the 008 tag position 011 in all records specified in the **input file**.

11.3.1.4 Add Default Value for 040 Tag Subfield f (-f)

Description: Specifies that the system should add an 040 tag subfield f with a specified default value.

Note: This option is only valid if the **-i** option is set to z.

Format: `-f`

Example: If you want the default value for the 040 tag subfield f to be "amim", type:

```
UpdateAuthoritySubj.exe -iz -famim < input file > output file
```

The program will use "amim" as the default character value for the 040 tag subfield in all records specified in the **input file** when the **-i** option is set to z.

11.3.1.5 Use Bibliographic Record to Set Values. (-b)

Description: Specifies whether the first associated bibliographic record will be used to set authority record values. The second indicator of the bibliographic record's subject tag will be used to set the 008 tag position 011. In addition, if the 008 tag position 011 is 7, the subfield $2 will be copied to 040 tag subfield $f.

Two values determine the action to take if there are NO associated bibliographic records and this option is used:

- **0** = Output Authority IDs only if there are no bibliographic records attached.
- **1** = Use default values to update authority records.

Default: 0

Format: `-b`

Example: If you want to use the first associated bibliographic record to set authority values and, for records with no associated bibliographic records, output Authority IDs only, type:

```
UpdateAuthoritySubj.exe -b0 < input file > output file
```

The program will use the first associated bibliographic record to set authority values in all records specified in the **input file** and, for authority records with no associated bibliographic records, output the Authority IDs to the specified **output file**.

11.3.1.6 Update Authority Record Regardless Of Whether or Not Current Value is Valid (-U)

Description: Specifies that authority records will be updated regardless of whether or not their current 008 tag position 011 values are valid.

Format: `-U`

Example: If you want the system to update authority records regardless of whether or not their current values are valid, type:

```
UpdateAuthoritySubj.exe -U < input file > output file
```

The program will update all authority records specified in the **input file** regardless of the validity of their current values.

11.4 Creating Indexes and Duplicate Subject Headings

Important: Before you take this step, make sure you have completed the first four steps described in the section "Overview of Steps for Enabling Multiple Subject Headings" of this guide:

1. Define subject headings to be indexed
2. Run **CreateSubjectThesauri_1.sh**
3. Run **UpdateAuthoritySubj.exe**
4. In the Profiler, enable Multiple Subject Headings

After you have fixed any problems with authority records and enabled Multiple Subject Headings in the Profiler, you will be ready to run the script **CreateSubjectThesauri_2.sh** against the database. This script…

- Uses the file of bib-ids generated in **CreateSubjectThesauri_1.sh** to create duplicate subject headings for subject headings in more than one thesaurus.
- Creates indexes for each thesaurus.

To create indexes and duplicate subject headings,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type:

   ```
   CreateSubjectThesauri_2.sh
   ```

3. Press Enter.

The program creates indexes for each thesaurus and creates duplicate subject headings for subject headings in more than one thesaurus.

12. Working with Linked and FRBR Records

This section describes scripts that perform tasks related to linked and FRBR records.

This chapter covers the following topics:

12.1 Masking and Unmasking Linked Records

If your institution has a large number of child records, you may wish to mask or unmask them in a batch process. The script **SetMaskAndKeywordIndexBibs.sh** masks or unmasks child records based on a text file you create, which contains the bibliographic IDs of the parent records. The script also reindexes these records for keyword searching immediately after masking or unmasking them.

Masked records affect the display of records in almost every Virtua subsystem, but you can find some general information about them in the *Virtua OPAC User's Guide*. For more information on masked records in a particular module, see the Virtua user's guide for the subsystem in which you are interested.

12.1.1 Running SetMaskAndKeywordIndexBibs.sh

To mask or unmask records and reindex them for keyword searching, run the script **SetMaskAndKeywordIndexBibs.sh**. This script runs **SetMaskChildren.sql** to mask or unmask a file of bibliographic records, and it runs **KeywordIndex.exe** to reindex the records for keyword searching. For information about **KeywordIndex.exe**, see the *System Management: OPAC User's Guide*.

SetMaskAndKeywordIndexBibs.sh uses an input file of records. The input file of records must be a text file that resides in the database directory. This text file must contain the bibliographic IDs of the parent records of any child records you wish to mask or unmask, with one ID per line.

To run SetMaskAndKeywordIndexBibs.sh,

1. Log in to your server as **dbadmin.**

2. At the prompt, type:

   ```
   SetMaskAndKeywordIndexBibs.sh [filename][maskflag]
   ```

   Where...

   - **`[filename]`** is the name of your input file, and

   - **`[maskflag]`** defines whether the records are going to be masked or unmasked. The valid values are:

     - **`0`** - **SetMaskAndKeywordIndexBibs.sh** will unmask the children records of the specified bibliographic IDs.
     - **`1`** - **SetMaskAndKeywordIndexBibs.sh** will mask the children records of the specified bibliographic IDs.

After you enter the maskflag value, **SetMaskAndKeywordIndexBibs.sh** will mask or unmask the children records according to your choice and update your keyword table to reflect the current mask status for the updated records in your database.

12.2 Removing Circular Links from Parent/Child Records

Virtua prevents you from saving a bibliographic record that would cause a "circular link" between parent/child records (i.e., two records with references to each other in their 004 tags). The software also prevents you from saving a bibliographic record that contains a circular link within a single record (i.e., matching 001 tag and 004 tag data in a single record).

The script **DeleteCircularParentChildLinks.sh** allows you to fix existing circular links between parent/child records (or within single records) that may already be present in the database.

To run DeleteCircularParentChildLinks.sh,

1. Log in to your server as **dbadmin**.
2. At the database prompt, type:

   ```
   DeleteCircularParentChildLinks.sh
   ```

3. Press Enter.

   The script deletes entries in the bibliographic_parent_child database table that are circular links.

12.3 Managing FRBR Records

Two scripts can help you manage the FRBR records in your database: **FindMissingFRBRLinks.sh** and **CheckAllFRBRLinks.sh**.

12.3.1 Identifying FRBR Records with Missing Parent Links

The script **FindMissingFRBRLinks.sh** identifies FRBR bibliographic records that have missing parent links.

To run FindMissingFRBRLinks.sh,

1. Log in to your server as **dbadmin**.
2. At the database prompt, type:

   ```
   FindMissingFRBRLinks.sh
   ```

3. Press Enter.

   The script identifies FRBR bibliographic records that have missing parent links, and logs the Bib-ID of each record that is found.

12.3.2 Checking All FRBR Links

The script **CheckAllFRBRLinks.sh** checks all records found in the frbr_parent_child database table and verifies that they are linked to the correct type of FRBR entity (Work, Expression, or Manifestation).

To run CheckAllFRBRLinks.sh,

1. Log in to your server as **dbadmin**.
2. At the database prompt, type:

   ```
   CheckAllFRBRLinks.sh
   ```

3. Press Enter.

   The script identifies FRBR records that are incorrectly linked, and logs the Bib-ID of each record that is found. If non-FRBR bibliographic records are found in the frbr_parent_child table, the script deletes them and logs the number of records deleted.

13. Identifying, Modifying and Removing Characters

This section describes programs that change the characters within the records in your database.

This chapter covers the following topics:

13.1 Normalizing Unicode Characters

The script **Re_indexForUnicodeNormalization.sh** lets you normalize Unicode characters in your bibliographic and authority records using the Unicode Standard Normalization Form C (NFC), and re-index the records. For more information about Unicode normalization in general and this normalization form in particular, see http://www.unicode.org/reports/tr15/.

This script will only be necessary for users who have been using Virtua for some time and have experienced problems when searching for or viewing Unicode data. If you are unsure about whether you should run this script, contact an Innovative Customer Support representative.

It is recommended that you cease all cataloging activity before running this script.

To normalize Unicode characters in your database:

1. Log in to your server as the **dbadmin** user.

2. At the prompt, type:

```
Re_indexForUnicodeNormalization.sh
```

3. Press Enter.

Virtua re-indexes all of the bibliographic and authority records in your database, normalizing Unicode characters according to Unicode Standard Normalization Form C (NFC). The script will output to the screen a confirmation message when the script completes. Note that it may take up to a few hours for this script to complete.

13.2 Removing Invalid UTF-8 Characters in Records

Virtua provides the scripts **RepairUTF8InAuthorityRecords.sh** and **RepairUTF8InBibRecords.sh,** which remove invalid UTF-8 characters, including the Unicode Replacement Character U+FFFD (efbfbd), FFFE and FFFF, from authority records and bibliographic records in the database. The scripts also check for the high and low surrogate ranges (0xD800 - 0xDFFF).

In addition, Virtua provides the executable **RepairUTF8InBibRecords.exe**, which you can run with command-line options to remove or only report on invalid UTF-8 characters from a file of MARC bibliographic records or Bib IDs used as standard input.

Note: The Unicode Replacement Character is used to replace an incoming character whose value is unknown or cannot be represented in Unicode.

13.2.1 Removing Invalid Characters in Authority Records

To remove invalid UTF-8 characters from authority records in the database, run **RepairUTF8InAuthorityRecords.sh**.

To run RepairUTF8InAuthorityRecords.sh,

1. Log in to your server as **dbadmin**.
2. At the database prompt, type: **`RepairUTF8InAuthorityRecords.sh`**
3. Press Enter.

The script runs and does the following:

- Creates a file of all Auth-IDs in the database.
- Using the file of Auth-IDs as input, runs **RepairUTF8InAuthorityRecords.exe** to correct the authority records with invalid characters.
- Writes the corrected authority records to a file.
- Runs **vload.exe** to reload the corrected authority records to the database.

13.2.2 Removing Invalid Characters in Bibliographic Records

To remove invalid UTF-8 characters from bibliographic records in the database, run **RepairUTF8InBibRecords.sh.**

To run RepairUTF8InBibRecords.sh,

1. Log in to your server as **dbadmin**.
2. At the database prompt, type: **`RepairUTF8InBibRecords.sh`**
3. Press Enter.

   The script runs and does the following:

   - Creates a file of all Bib-IDs in the database.
   - Using the file of Bib-IDs as input, runs **RepairUTF8InBibRecords.exe** to correct the bibliographic records with invalid characters.
   - Writes the corrected Bib-IDs to a file.
   - Using the file of corrected Bib-IDs as input, runs **MoveStateRecords.exe** to reprocess (catalog) the bibliographic records.

13.2.3 Removing or Reporting on Invalid Characters in a File of MARC Bibliographic Records or Bib IDs

To remove or report on invalid UTF-8 characters from a file of MARC bibliographic records or Bib IDs used as standard input, run **RepairUTF8InBibRecords.exe**. Invalid UTF-8 characters include the Unicode Replacement Character U+FFFD (efbfbd), FFFE and FFFF. The program also checks the high and low surrogate ranges (0xD800 - 0xDFFF).

The program allows for two command-line options. The basic format for running **RepairUTF8InBibRecords.exe** is:

RepairUTF8InBibRecords.exe [command options] < [input file] > [output file]

Where . . .

[command options] are the allowed options for specifying the following:

- **-i [--input-marc-records]** – Input a file of MARC bibliographic records. (Default=false, in which case the file will consist of Bib IDs.)

- **-r [--report-only]** – Report on invalid characters but do not update the MARC bibliographic records. (Default=false, in which case the program updates the records in the database.)

- **-b [--outputFileForBibIDs]** – When the **-b** option is used in conjunction with the **-i [--input-marc-records]** option, the program outputs a file of the Bib IDs of the records found to contain invalid UTF-8 characters.

- **-? [--help]** – View options for the executable.

To run RepairUTF8InBibRecords.exe,

1. Log in to your server as **dbadmin**.

2. At the database prompt, type:

   ```
   RepairUTF8InBibRecords.exe [command options] < [input file] >
   [output file]
   ```

3. Press Enter.

   The program will perform according to the following scenarios:

- If **-i** and **-r** are set, the input will be a file of MARC records, and the output will be a file of the *unchanged* MARC records that have invalid UTF-8 characters.

- If **-i** is set without **-r**, the input will be a file of MARC records, and the output will be a file of corrected (updated) MARC records.

- If **-i** is NOT set, the input will be a file of Bib IDs, and the output will be a file of the Bib IDs of the records that contain invalid UTF-8 characters. Whether the file contains the Bib IDs of records that are unchanged or updated in the database depends on whether **-r** is set.

- If **-r** is set, the output will be a file of unchanged MARC records or a file of Bib IDs of unchanged MARC records, depending on whether **-i** is set.

- If **-i** and **-b** are set, the program outputs a file of the Bib IDs of the records found to contain invalid UTF-8 characters.

13.2.4 Removing Invalid Characters in a File of XML Records before Loading

To remove invalid UTF-8 characters and control characters from a file of XML records used as standard input, run **RepairXMLFile.exe**.

Before loading XML records with **vload.exe**, pre-process all XML files using this program.

To run RepairXMLFile.exe,

1. Log in to your server as **dbadmin**.

2. At the database prompt, type:

   ```
   RepairXMLFile.exe < [input file] > [output file]
   ```

3. Press Enter.

13.3 Removing <CR><LF> Characters from a File of MARC Records

Virtua provides an executable that removes unwanted control characters from a file of MARC records so that they will be "clean" for loading into the database. **RemoveCRLF.exe** removes the combination of <CR><LF> characters from a file of incoming MARC records.

To run RemoveCRLF.exe,

1. Log in to your server as **dbadmin**.

2. At the database prompt, type:

   ```
   RemoveCRLF.exe < [input file of records] > [output file of records]
   ```

3. Press Enter.

   The program runs and reads each MARC record from stdin and replaces the combination of <CR><LF> characters with a single space.

13.4 Updating Old Ayn and Alif Characters

If the database of records you import contains the old Unicode characters Ayn, U+02BF, or Alif, U+02BE, you can update these characters using the scripts **MapAynAlifCharsInAuthorityRecords.sh** to update your Authority records and **MapAynAlifCharsInBibRecords.sh** to update your bibliographic records. These scripts will update all U+02BF characters to the newer U+02BB characters, and update U+02BE characters to U+02BC characters.

Note: If you do not run these scripts and you have a mix of older and newer Ayn and Alif chracters in your records, Virtua will not be able to group these records together when sorting or indexing the records.

13.4.1 Updating Ayn and Alif Characters in Bibliographic Records

If you have Ayn or Alif characters in bibliographic records in your collection and you would like to update these records, you can use the script **MapAynAlifCharsInBibRecords.sh** to update all U+02BF characters to U+02BB characters, and all U+02BE characters to U+02BC characters.

To run MapAynAlifCharsInBibRecords.sh,

1. Log in to your server as the **dbadmin** user.

2. At the prompt, type:

```
MapAynAlifCharsInBibRecords.sh
```

This script will automatically update all occurrences of U+02BF characters to U+02BB characters, and all U+02BE characters to U+02BC characters in the bibliographic records in the database.

13.4.2 Updating Ayn and Alif Characters in Authority Records

If you have Ayn or Alif characters in authority records and you would like to update these records, you can use the script **MapAynAlifCharsInAuthorityRecords.sh** to update all U+02BF characters to U+02BB characters, and all U+02BE characters to U+02BC characters.

To run MapAynAlifCharsInAuthorityRecords.sh,

1. Log in to your server as the **dbadmin** user.

2. At the prompt, type:

```
MapAynAlifCharsInAuthorityRecords.sh
```

This script will automatically update all occurrences of U+02BF characters to U+02BB characters, and all U+02BE characters to U+02BC characters in the authority records in the database.

13.5 Changing ß Characters

If the database of records you import contains the Unicode character **ß**, U+00DF, and you want to change this to the standard **ss**, U+0073 U+0073, you can update these characters using the program **MapSharpSCharacters.exe** to update your MARC records and **MapSharpSCharactersInAuthorityRecords.sh** to update your authority records. These scripts will update all U+00DF characters to two U+0073 characters in a file of records; i.e, each instance of **ß** is changed to **ss**.

13.5.1 Changing ß Characters in MARC Records

The executable **MapSharpSCharacters.exe** processes a file of MARC 2709 records to replace all instances of the **ß** character to **ss**.

To change each instance of the ß character in a file of MARC records,

Upload to the server a file of MARC records containing the ß character.

1. Log in to your server as the **dbadmin** user.

2. Upload to the server a file of MARC records to process.

3. At the dbadmin prompt, type:

   ```
   MapSharpSCharacters.exe < inputfile > outputfile
   ```

   Where **inputfile** is the name of the file of MARC records containing the **ß** character, and **outputfile** is the name of the file of corrected records for the program to create.

Virtua creates a file of records where each instance of the **ß** character is changed to **ss**. You can load this file of modified records into your database. See the *Virtua Record Loading User's Guide* for more information.

13.5.2 Changing ß Characters in Authority Records

The script **MapSharpSCharactersInAuthorityRecords.sh** changes all of the authority records in your database such that all instances of the **ß** character are replaced by **ss**.

To change all instances of ß in a file of authority records,

1. Log in to your server as the **dbadmin** user.

2. At the dbadmin prompt, type:

```
MapSharpSCharactersInAuthorityRecords.sh
```

Virtua performs a number of steps, including running **MapSharpSCharacters.exe**, to change all instances of the **ß** character in authority records in your database to **ss**.

13.6 Identifying Bibliographic Titles Containing Return Characters

The script **IdentifyBibTitleWithReturnChars.sh** identifies all bibliographic records that have Microsoft Windows return (CR) characters in their title and writes the Bib-ID of each record and the tag with the return characters to a file. CR characters are carriage returns followed by a line feed. Return characters in titles cause problems in SIP messaging, i-tiva notices, and InfoStation i-tiva reports. The resulting file of Bib-IDs lets you access the bibliographic records via the Virtua client and remove the return characters in the title tags.

Usage: `IdentifyBibTitleWithReturnChars.sh`

The script outputs a comma-delimited text file, **BibsWithReturnChar.txt**, which contains a list of Bib-IDs and the tag(s) that contains a return character.

13.7 Removing Tab Characters from Bibliographic Records

The executable **RemoveTabCharacterFromBibRecords.exe** removes tab characters before and after subfields in tags of MARC bibliographic records and generates a report of the fixes.

To run RemoveTabCharacterFromBibRecords.exe,

1. Log in to your server as **dbadmin**.

2. At the database prompt, type:

```
RemoveTabCharacterFromBibRecords.exe [options] <inputFile
>outputFile
```

Where…

[options] consist of the following:

-? [--help] - Print usage statement.
-r [--input-marc-records] - Input MARC records. (Default = false)
-t [--report-tags-only] - Only report tags containing tabs; do not update. (Default=false)

-AND-

inputFile = The name of a file of Bib IDs.
-OR-
If **-r [--input-marc-records]** is used, the name of a file of MARC records.

outputFile = The name of a file of updated Bib IDs.
-OR-
The name of a file of MARC records with tab characters removed.

Note: If **-t [--report-tags-only]** is used, no output file will be created and no updates will be made.

3. Press Enter.

Example 1:

```
$EXE_DIR/RemoveTabCharacterFromBibRecords.exe < bibids >
bibidsfixed
```

This format removes tab characters from tags in database records by the Bib IDs that are contained in the file **bibids** and writes the Bib IDs of only the fixed records to the file **bibidsfixed**.

The report that is generated lists the Bib ID of each record that was fixed. Each Bib ID serves as the record number and is listed along with the tags affected by the fix.

Example 2:

```
$EXE_DIR/RemoveTabCharacterFromBibRecords.exe --input-marc-records
< Bella.rec > BellaFixed.rec
```

This format removes tab characters from tags in the MARC bibliographic records in the file **Bella.rec** and writes ALL of the records, in addition to the fixed records, to a new file **BellaFixed.rec**.

The report that is generated lists each record that was fixed. The records are numbered based on the position of each in the input file; thus, #1, #2, #3, etc.
.

14. Patron-related Scripts

This chapter covers the following topics:

You can find information about other scripts related to patron records in the sections "Extracting Patron Records by Date Range," "Extracting Error State Patron Records," and "Extracting Duplicate Patron Records" in this guide.

14.1 Loading Postal Codes into Virtua

The script **PopulatePostalCode.sh** lets you load a file of postal codes into Virtua, which enables Virtua to automatically populate the city and other fields when a postal code is entered in the Patron Editor in the Virtua client. This also automatically populates the Postal Code editing tool in the Virtua Profiler. For information about the Postal Code editing tool in the Virtua Profiler, see the *Virtua Profiler/Global Settings User's Guide*. For information about the Patron Editor, see the *Virtua Circulation/Patron Information User's Guide*.

14.1.1 Preparing Postal Code Data for Virtua

Virtua comes with a complete file of U.S. postal codes that you can load into Virtua. To load these default postal codes, see the section "Loading Virtua's Default List of Postal Codes" below.

Alternatively, you can upload your own file of postal codes that are relevant for your library patrons. The file of postal codes *must* be a **.dat** file with comma-delimited data in the following format:

```
Postal code,City,State,County,Country
```

Only the postal code and city are required fields. Each entry is on its own line. If a given field does not exist for an entry, you can skip the entry.

Example entries:

All fields:	`24060,Blacksburg,Virginia,Montgomery, United States`
Missing County:	`24060,Blacksburg,Virginia,,United States`
Missing County, State:	`24060,Blacksburg,,,United States`
Postal Code and City only:	`24060,Blacksburg`

Note: There should be no spaces between the fields, though there can be spaces within each field.

Once you create a file of postal codes, save the file in **.dat** format to your Virtua server, and use **PopulatePostalCode.sh** to load your file into Virtua, as described below.

14.1.2 Running PopulatePostalCode.sh

You can run **PopulatePostalCode.sh** with several command-line options. You can choose to run the script to . . .

- Load Virtua's default list of U.S. postal codes
- Load your own ***.dat** file of postal codes
- Delete postal codes currently loaded in Virtua and load your own ***.dat** file of postal codes.

We describe the process for each of these options in the following sections.

14.1.2.1 Loading Virtua's Default List of Postal Codes

To run PopulatePostalCode.sh to load Virtua's default list of postal codes,

1. Log in to your server as **dbadmin**.
2. At the database prompt, type:

   `PopulatePostalCode.sh`

3. Press Enter.

The script asks you which file of postal code data you wish to load from Virtua's default list of U.S. postal codes.

4. Specify your desired file of postal codes and press Enter.

The script loads the specified postal codes.

14.1.2.2 Loading a File of Postal Codes

To run PopulatePostalCode.sh to load your own file of postal codes,

1. Log in to your server as **dbadmin**.
2. At the database prompt, type:

   ```
   PopulatePostalCode.sh [filename]
   ```

   Where **[filename]** is the name of your **.dat** file of postal codes. If your file is not in the current directory, you *must* include the full path to your file.

3. Press Enter.

The script loads your file of postal codes into Virtua.

Note: For information on the correct format for your file of postal codes, see the section "Preparing Postal Code Data for Virtua" in this guide.

14.1.2.3 Deleting a File of Postal Codes and Loading a File of Postal Codes

To run PopulatePostalCode.sh to load a file of postal codes and delete currently loaded postal codes,

1. Log in to your server as **dbadmin**.
2. At the database prompt, type:

   ```
   PopulatePostalCode.sh -d [filename]
   ```

   Where **[filename]** is the name of a **.dat** file of postal codes that you would like to upload. If your file is not in the same directory as the **PopulatePostalCode.sh** script, you must include the full path to your file.

3. Press Enter.

The script deletes the existing postal code data and saves the new postal code data, which will be available when the Patron Editor is used to create or modify a patron street address.

Note: For information on the correct format for your file of postal codes, see the section "Preparing Postal Code Data for Virtua" in this guide.

14.2 Loading Patron Languages into Virtua

The script **LoadPatronLanguageCode.sh** lets you load a file of language codes into the database, thus providing the list of languages on the Patron Language Codes window in the Virtua Profiler. This Profiler parameter defines which languages will appear as Language Preference choices when creating or updating a patron record via the Patron Editor.

This script is run each time Virtua is updated, so you only need to run this script if you wish to make a change to the Patron Language Codes available in the Virtua Profiler.

LoadPatronLanguageCode.sh takes an input file name as a command-line option. The file can be any type of UNIX text file, but the contents must be a line-delimited file in the format:

LanguageName;LanguageCode

Example:
English;en
Swiss French;fr-ch
Deutsch;de

To run LoadPatronLanguageCode.sh to load Virtua's default list of postal codes,

1. Log in to your server as **dbadmin**.

2. At the database prompt, type:

```
LoadPatronLanguageCode.sh [filename]
```

Where **[filename]** is the name of a text file of language codes.

3. Press Enter.

Any existing patron languages in the database will be overwritten with the languages in the text file. If there is a problem with the input file, the errors will be saved to the **Patron_Language_Code_Table.log** file and the existing patron languages will be maintained.

If no input file name is provided, the file **$EXE_DIR/PatronLanguageCodes.dat** will be used.

15. Appendix A - Table of Scripts and Executables Discussed in this Guide

The following table lists and briefly describes each script and executable discussed in this guide. For additional information on each program, click the name in the first column, which is hyperlinked to the appropriate section in this guide.

Program name	Purpose	Command-line parameters
AddItemCallNumberPrefix.sh	Adds a prefix to the call numbers of the specified item records.	This script accepts . . . • The user ID to be used as the operator for processing • The prefix to add to the item-level call numbers • The name of a file that contains a list of item IDs.
addItemStatus.sh	Adds an item status to the specified records.	This script accepts . . . • The name of a file that specifies the item barcodes of the items to which you want add statuses and the status that you want to add. • The location *code* of the location for which you want to make modifications.
AddSpecial035.exe	Adds a special 035 tag to bibliographic records for use in duplicate checking. For use by select customers only.	See the *Virtua Record Loading User's Guide* for details.

Program name	Purpose	Command-line parameters
authClr.sh	Removes orphaned headings from the browse list.	None
BatchDeleteItems.exe	Deletes the item records associated with the specified list of Item-IDs.	This executable accepts one command-line option (check unserviceable) and the name of an input file of Item-IDs.
ChangeAllowRequest.sh	For a specified set of item records, sets the Allow Request flag to the specified value.	This script accepts . . . • The Allow Request value that should be assigned to the specified items. • A value that specifies the type of input. • The name of the input file.
ChangeItemClass.sh	Evaluates a list of items and updates any "old item class" it finds with the "new item class."	This script accepts . . . • The item class code of the "old item class" that you want to change. • The item class code of the "new item class" with which to replace the old item class. • A value that specifies the type of input. • The name of the input file.
ChangeItemPrice.sh	Changes an item price to a different item price in the specified records.	This script accepts . . . • The item price that you want to change. • The new item price with which to replace the existing item price. • A value that specifies the type of input. • The name of the input file.

Program name	Purpose	Command-line parameters
ChangeItemStatus.sh	Changes an item status to a different item status in the specified records.	This script accepts . . . • The name of a file that specifies the item barcodes of the items for which you want to change a status. • The location code of the location for which you want to make these modifications. • The code of the status that you want to change. This status will be removed from each item. • The code of the status that you want to add to records that have the status defined by the previous parameter.
ChangeLoanPeriod.sh	Changes the loan period in the specified records.	This script accepts . . . • The new loan period flag that you want to assign to items. • The existing loan period value that you want to update. • The new loan period value that should be assigned to items that have the specified existing value. • A value that specifies the type of input. • The name of the input file.

Program name	Purpose	Command-line parameters
ChangeLocation.sh	Modifies the owning location of items that you specify in a file of Item-IDs or barcodes.	This script accepts . . . • The location code of the "old location" that you want to change. • The location code of the "new location" that replaces the old location. • A value that specifies the type of input. • The name of the input file.
ChangeLocationByBarCode.sh	Modifies the location of items that you specify in a file of barcodes.	This script accepts . . . • The code for the location to which the items will be changed. • The name of a file of item barcodes that will be changed.
ChangeLocationByCallNumberRange.sh	Modifies the shelf and owning location of items within a range of item-level call numbers.	This script provides interactive prompts.
ChangeLocationByItemId.sh	Modifies the location of items that you specify in a file of Item-IDs.	This script accepts . . . • The code for the location to which the items will be changed. • The name of a file of Item-IDs that will be changed.
ChangeLocationId.sh	Changes one location code to a new location code and updates all the database tables that use it.	This script accepts . . . • The location code of the "old location" that you want to change. • The location code of the "new location" that replaces the old location.
ChangePermAuthorityToProv.sh	Converts permanent authority records to provisional.	This script accepts the name of a file of record IDs for the authority records to be converted.

Program name	Purpose	Command-line parameters
ChangeShelfLocation.sh	Modifies the owning location of items that you specify in a file of Item-IDs or barcodes.	This script accepts . . . • The location code of the "old location" that you want to change. • The location code of the "new location" with which to replace the old location. • A value that specifies the type of input. • The name of the input file.
ChangeReserveItemClass.sh	Evaluates a list of items and updates any "old reserve item class" it finds with the "new reserve item class."	This script accepts . . . • The item class code of the "old reserve item class" that you want to change. • The item class code of the "new reserve item class" with which to replace the old item class. • A value that specifies the type of input. • The name of the input file.
CheckAllFRBRLinks.sh	Checks all records found in the frbr_parent_child database table and verifies that they are linked to the correct type of FRBR entity (Work, Expression, or Manifestation).	None
CheckForReciprocal1xx5xxAuthorities.sh	Checks in authority records for "un-linked" 5xx cross-references (i.e., a 5xx tag *See also* heading that is not linked to a 1xx tag in an LC authority record). The script creates an output file of authority records found to have "un-linked" 5xx headings.	This script accepts a start date and an end date.

Program name	Purpose	Command-line parameters
ConvertMARCFileCharacterSet.exe	Converts a file of MARC records from one character set to UTF-8 or vice versa.	This executable accepts the code of the source character set and that of the destination character set.
ConvertMARCRecordsToRDA.exe	Provides numerous options for converting a file of MARC authority or bibliographic records to RDA.	This executable accepts the name of an input file of MARC records and numerous options.
ConvertMARCRecordsToRDA.sh	Calls a helper file to load RDA translations into the database and then converts a file of MARC records to RDA.	This script accepts the name of an input file of MARC records and three optional input parameters.
CreateBibIdsFromItemIds.sh	Generates a file of Bib IDs from an input file of Item IDs.	This script accepts the name of an input file of Item IDs.
CreateItemBarcodesFromItemIds.sh	Generates a file of item barcodes from an input file of Item IDs	This script accepts the name of an input file of Item IDs.
CreateItemIdsFromBarcodes.sh	Generates a file of Item IDs from an input file of item barcodes.	This script accepts the name of an input file of item barcodes.
CreateItemIdsFromBibIds.sh	Generates a file of Item IDs from an input file of Bib IDs.	This script accepts the name of an input file of Bib IDs.
CreateSubjectThesauri_1.sh	Identifies any problems with authority records as part of the enable multiple subject headings process.	None

Program name	Purpose	Command-line parameters
CreateSubjectThesauri_2.sh	As part of the enable multiple subject headings process… • Uses the file of bib-ids generated in **CreateSubjectThesauri_1.sh** to create duplicate subject headings for subject headings in more than one thesaurus. • Creates indexes for each thesaurus.	None
DeleteCircularParentChildLinks.sh	Deletes existing circular links (i.e., two records with references to each other in their 004 tags) between parent/child bibliographic records	None
DeleteItemsByBarcode.sh	Deletes the item records associated with the specified list of item barcodes.	This script accepts the name of an input file of item barcodes.
DeleteItemsByDueDate.sh	Deletes from the database all items that were due (and still checked-out) *before* the specified date.	This script accepts a due date and optionally a location code.
DeleteItemsByLocation.sh	Deletes the item records associated with the specified location code.	This script accepts as input the location code of the location whose items you want to delete.
DeleteItemsByStatus.sh	Deletes from the database all items that carry the specified status.	This script accepts a status code and optionally a location code.
extract_bibs_by_creation_date.sh	Extracts bibliographic records based on creation date.	None

Program name	Purpose	Command-line parameters
extract_deleted_bibs.pl	Extracts MARC 21 (UTF-8) bibliographic records in a Virtua database that are in Delete state.	This script accepts the following command line options: [--start-date "YYYY-MM-DD HH:MI:SS"] [--end-date "YYYY-MM-DD HH:MI:SS"] [--no-status-change] [--help]
extract_patrons_by_modify_date.sh	Writes a file of patron records that have been modified within a specified date range.	None. This is an interactive program.
extract_patrons_in_error_state.pl	Extracts patron records in a Virtua database that are in Error state.	This script accepts the following command line options: [--start-date "YYYY-MM-DD HH:MI:SS"] [--end-date "YYYY-MM-DD HH:MI:SS"] [--extract-and-delete] [--delete-only] [--help]
ExtractAuthorityRecordsUsingControlNumbers.sh	Connects to a Virtua database and creates a file of authority records from a file of authority control numbers.	None.
ExtractForDiscovery.sh	Extracts bibliographic records, either fully or incrementally, for discovery purposes and converts them to XML format.	This script accepts a single number, 0 or other, representing number of days.
FindMissingFRBRLinks.sh	Identifies FRBR bibliographic records that have missing parent links.	None

Program name	Purpose	Command-line parameters
FixAuthConflictErrors.sh	Changes the 4xx tag in authority records with a specified prefix by adding a subfield $z to the 4xx tag that contains the 035 control number of the authority record.	This script accepts the following command line options: [prefix] [filename]
globalChange1.ksh	Outputs a list of authority records containing the text you specify.	This script accepts a code representing the authority record type and a string to search for in each authority record of that type.
globalChange2.ksh	Finds and replaces text in a tag range of a given list of authority records.	This script accepts the following command line options: • A code representing the range of tags in which you want to search and replace. • The text you want to replace. • The replacement text. • The name of the file listing authority IDs.
IdentifyBibTitleWithReturnChars.sh	Identifies and writes to a file a list of Bib-IDs and title tags in bibliographic records that contain a Windows return character.	None
IdentifyDuplicatePatronBarcodes.sh	Identifies and writes to a file a list of duplicate patron barcodes.	None
ImportRDATranslations.sh	A helper script that can be used in standalone mode to load either bibliographic or authority RDA translations into the database (see **ConvertMARCRecordsToRDA.sh**).	This script accepts *Auth* or *Bib* as a required parameter.

Program name	Purpose	Command-line parameters
ItemStatusMonitor.exe	Finds any items with a status that has expired and moves it to the next status. For information about item statuses, see the *Virtua Profiler/ Global Settings User's Guide.*	**--interval** [interval] Where [interval] is the interval at which you want the program to check for item statuses that need to be updated.
List853Holdings.sh	Extracts the 853 tags from the holdings records in your database.	None
LoadPatronLanguageCode.sh	Loads the languages in a text file into the database to be used as possible patron languages.	This script accepts the name of a file of languages and language codes.
LoadRecordConverterFiles.sh	Places and stores a style sheet for MODSXML format conversions on the server	None
MapAynAlifCharsInAuthorityRecords.sh	Updates existing Unicode U+02BF characters in authority records to the updated U+02BB characters, and updates U+02BE characters to U+02BC characters.	None
MapAynAlifCharsInBibRecords.sh	Updates existing Unicode U+02BF characters in bibliographic records to the updated U+02BB characters, and updates U+02BE characters to U+02BC characters.	None
MARCBibUpdate.exe **Important:** Do NOT use this executable until you read the chapter "Modifying Bibliographic Data."	Alters, in batch, bibliographic records in a database or in a file based on the input options you specify.	See the section "Command Line Reference for MARCBibUpdate.exe"

Program name	Purpose	Command-line parameters
MarcView.exe	Outputs records in an easy-to-read format.	This executable accepts the following command line options. • **-s** - Indicates what character will be used as the subfield code in the output. • **-T** - Indicates the type of MARC record. • **-w** - Indicates the source of the records that will be extracted.
MoveStateRecords.exe **Important:** Cease all cataloging activity before running this program.	Processes records and attempts to move them to a new state.	None - This is an interactive program. The executable will prompt you for required information.
PermAuthIdsToProv.ksh	Creates a file of IDs for permanent authority records that are candidates for conversion back to provisional. Records that qualify have a 1XX tag but no cross-references (4XX, 5XX, or 7XX) and no control numbers (010 or 035 tags).	This script accepts the name of a file to which it will write the ID of records that are candidate for conversion.
PopulateBibFields.exe	Populates the bibliographic_fields table in the database with bibliographic record data.	This executable requires the name of a file of Bib IDs.
PopulateBibOtherIdentifier.sh	Populates the bib_other_identifier table in the database with bibliographic 024 tag data.	None
PopulateBibPublisherNumber.sh	Populates the bib_publisher_number table in the database with bibliographic 028 tag data.	None

Program name	Purpose	Command-line parameters
PopulatePostalCode.sh	Loads postal codes into Virtua or deletes existing postal codes.	This script accepts the following command-line options: • **[filename]** - Loads the specified **.dat** file of postal codes into Virtua. • **-d [filename]** - Deletes existing postal codes and loads the specified **.dat** file of postal codes into Virtua.
PopulateUDCBrowse.sh	Populates the UDC Browse table with new rules as defined in the Virtua Profiler so a user can browse by UDC.	None
PrintAuthoritybyState.ksh	Writes to a file the authority IDs of every record in the database with a given state. The resulting file will be sorted by bib level, tag, and heading.	None
RecordConverter.exe	Converts a file of records that are in one of the following three formats—ISO-2709, MARCXML or MODSXML—to a file of records in either ISO-2709 format or MARCXML format	This executable requires the input and output record formats and the input and output filenames.

Program name	Purpose	Command-line parameters
removeAddItemStatus.ksh	Removes an item status from and adds an item status to the specified records.	This script accepts . . . • The name of a file that specifies the item barcodes of the items for which you want to make modifications. • The location code of the location for which you want to make these modifications. • The code of the status that you want remove from each record. • The code of the status that you want to add to each record.
RemoveCRLF.exe	Removes the <CR><LF> control character combination from a file of MARC records.	None
RemoveBlankItemCallNum.sh	Removes blank item-level call numbers, which can alleviate problems with **CallIndex.sh**.	None
removeItemStatus.ksh	Removes an item status from the specified records.	This script accepts the name of a file that specifies the item barcodes or item IDs of the items from which you want to remove a status and the status that you want to remove.
RemoveTabCharacterFromBibRecords.exe	Removes tab spaces from MARC bibliographic records and generates a report.	This script accepts the name of a file of Bib IDs or a file of MARC records from which you want to remove tab spaces.
RemoveXMLRecordFromBibs.sh	Removes XML data from bibliographic records in the database.	This script requires the name of a file of bibliographic IDs.

Program name	Purpose	Command-line parameters
RepairUTF8InAuthorityRecords.sh	Removes invalid UTF-8 characters, including the Unicode Replacement Character U+FFFD, from authority records.	None
RepairUTF8InBibRecords.exe	Removes invalid UTF-8 characters, including the Unicode Replacement Character U+FFFD, from MARC records.	This executable accepts as stdin a file of MARC records from which you want to remove invalid UTF-8 characters.
RepairUTF8InBibRecords.sh	Removes invalid UTF-8 characters, including the Unicode Replacement Character U+FFFD, from bibliographic records.	None
RepairXMLFile.exe	Removes invalid UTF-8 characters and control characters from a file of XML records used as standard input.	None
ReplaceSubstringInItemCallNumber.sh	Replaces a defined substring in item-level call numbers with another substring.	• The user ID to be used as the operator for processing • The string of characters to be removed from the call numbers • The string that will replace the removed string • The name of a file containing the item IDs of the items to have a substring replaced.
ReProcessAuths.ksh **Important:** Cease all cataloging activity before running this script.	Reprocesses all authority records and attempts to move them to the Process Immediately state.	This script accepts a file of record IDs for the authority records to be reprocessed.

Program name	Purpose	Command-line parameters
ReProcessBibs.ksh **Important:** Cease all cataloging activity before running this script.	Reprocesses all bibliographic records and attempts to move them to the Process Immediately state.	This script accepts a file of record IDs for the bibliographic records to be reprocessed.
ReProcessBibsToIndexVitalPid.sh	Creates a file of Bib IDs that contain tag 856 subfield $i VITAL and reprocesses the Bib IDs to index the VITAL PID in the Vital_Pid column of the Bibliographic_Record table.	None
ReProcessForPublisherHeadings.sh	Creates a file of Bib IDs that contain the 260 $b or 264 $b and reprocesses them to index the publisher headings for browse searches without updating the bibliographic records.	None
ReProcessForPublisherUserHeadings.sh	Creates a file of Bib IDs that contain the 260 $b or 264 $b and reprocesses them to index the publisher headings and user-defined headings for browse searches without updating the bibliographic records.	None
ReProcessPatrons.sh **Important:** Cease all patron cataloging activity before running this script.	Reprocesses all patron records and attempts to move them to the Process Immediately state	This script accepts a file of record IDs for the patron records to be reprocessed
rlint.exe	Prepares RLIN records for loading to the Virtua database.	This executable accepts as command line options the name of the file that contains the records to process and the name of the file to which the program will write the processed records.

Program name	Purpose	Command-line parameters
SetMaskAndKeywordIndexBibs.sh	Masks or unmasks a file of bibliographic records and reindexes the records for keyword searching.	This script accepts . . . • Name of text file containing parent bibliographic IDs • **0** - To unmask the children records of the specified bibliographic IDs. • **1** - To mask the children records of the specified bibliographic IDs.
SetPatronBarcodesAreUniqueFlag.sh	Sets a database parameter indicating that patron barcodes are unique across patron barcode types, i.e., across primary and alternate barcodes. When this flag is set, Virtua supports client request transactions using alternate patron barcodes. Virtua checks for the primary barcode first and if not found, checks for and uses the alternate barcode. (If there is a duplicate cross between the two barcode types, Virtua will consider *only* the primary barcode/tag 015 subfield $a when a request is placed.)	To set the database parameter and enable support for alternate patron barcodes in request transactions, run the script as follows: **`SetPatronBarcodesAreUniqueFlag.sh 1`**

Program name	Purpose	Command-line parameters
ToMARC21.exe	Converts UNIMARC authority records to MARC 21 format.	This executable accepts as command-line options the name of the file that contains the records to process and the name of the file to which the program will write the processed records. For information about piping the converted records directly to **vload.exe**, see the chapter "Converting UNIMARC Authorities to MARC 21" in this user's guide.
UnlockMARCRecord.sh	Unlocks a MARC record that is no longer being edited.	This script accepts the record ID, record type code, and (optionally) the "report_only" parameter.

Program name	Purpose	Command-line parameters
UpdateAuthoritySubj.exe	As part of the enable multiple subject headings process, fixes any problem authority records whose Authority IDs are provided in the input file	This executable accepts the following command line options. • **-R** - Specifies that no updates will occur. Authority IDs of invalid authority records will be output to the specified file. • **-D** - Specifies that the system should not check for associated bibliographic records and should instead use default values to update authority records. • **-U** - Specifies that authority records will be updated regardless of whether or not their current 008 tag position 011 values are valid. • **-i** - Specifies which character should be used as the default value for the 008 tag position 11. • **-f** - Specifies that the system should add an 040 tag subfield f with a specified default value (only valid if the -i option is set to z.). • **-b** - Specifies whether the first associated bibliographic record will be used to set authority record values.
vload.exe	Loads records to the database.	For information on **vload.exe**, see the *Virtua Record Loading User's Guide*.
write2709.exe	Extracts records from the database.	See the section "Command Line Reference for write2709.exe"
WriteAuthIdsFile.ksh	Extracts to a file a list of authority IDs for the authority records in your database.	This script accepts as a command line parameter the name of a file to which the script will write record IDs.

Program name	Purpose	Command-line parameters
WriteBibIdsFile.ksh	Extracts to a file a list of bibliographic IDs for the bibliographic records in your database.	This script accepts as a command line parameter the name of a file to which the script will write record IDs.
WriteHoldingIdsFile.ksh	Extracts to a file a list of holdings IDs for the holdings records in your database.	This script accepts as a command line parameter the name of a file to which the script will write record IDs.
WriteItemIdsFile.ksh	Extracts to a file a list of Item-IDs for the item records in your database.	This script accepts as a command line parameter the name of a file to which the script will write record IDs.
WriteMaskedBibIdsFile.sh	Extracts to a file a list of bibliographic IDs for the masked bibliographic records in your database.	This script accepts as a command-line parameter the name of a file to which the script will write record IDs.
WritePermAuthIds.ksh	Extracts to a file a list of authority IDs for the *permanent* authority records in your database.	This script accepts as a command line parameter the name of a file to which the script will write record IDs.
WritePermAuthorityFile.ksh	Using write2709.exe, extracts to a file the *permanent* authority records in your database.	This script accepts as a command line parameter the name of a file to which the script will write the records.
WritePermSubjectIds.ksh	Extracts to a file a list of authority IDs for the *permanent* subject authority records in your database.	This script accepts as a command line parameter the name of a file to which the script will write record IDs.
XMLToMARC.exe	Converts a file of MARCXML records to MARC 2709 records.	This executable requires as command-line parameters the input and output filenames.

16. Appendix B - HKPL-specific Script to Enable the HKPL Patron Editor

The script **SetHKPLFeatures.sh** can be run to enable the features in Virtua and Chamo that are specific to the Hong Kong Public Library. To enable the HKPL features, the script must be run with the command-line option of '1.'

17. Appendix C - Conversion Table: AACR2 to RDA

17.1 Mapping of AACR2 Cataloging Elements to RDA in MARC Records

17.1.1 Bibliographic Records

Mapping Current MARC Elements to New RDA Elements

Note: If any change (as described below, and including any change to an abbreviation) is made to the bibliographic record, **040$e rdax** is added to the record.

MARC	Current	New	Remarks
LDR 18	a	i	
245 $h			Map as shown below and then delete $h and data; lower or upper case is not important in the Current value
	Art original	336 $a still image	
		337 $a unmediated	
		338 $a other	
	Art reproduction	336 $a still image	
		337 $a unmediated	
		338 $a other	
	Braille	336 $a tactile text	
		337 $a unmediated	
		338 $a volume	
	Cartographic material	336 $a cartographic image	
		337 $a unmediated	
		338 $a other	
	Chart	336 $a text	
		337 $a unmediated	
		338 $a other	

MARC	Current	New	Remarks
	Diorama	336 $a three-dimensional form	
		337 $a unmediated	
		338 $a other	
	Electronic resource	336 $a text	
		337 $a computer	
		338 $a online resource	
	Filmstrip	336 $a still image	
		337 $a projected	
		338 $a filmstrip	
	Flash card	336 $a text	
		337 $a unmediated	
		338 $a card	
	Manuscript	336 $a text	
		337 $a unmediated	
		338 $a volume	
	Microform	336 $a text	
		337 $a microform	
		338 $a aperture card	if 007/01 is **a**
		338 $a microfiche	if 007/01 is **e**
		338 $a microfiche cassette	if 007/01 is **f**
		338 $a microfilm cartridge	if 007/01 is **b**
		338 $a microfilm cassette	if 007/01 is **c**
		338 $a microfilm reel	if 007/01 is **d**
		338 $a microfilm roll	if 007/01 is **j**
		338 $a microfilm slip	if 007/01 is **h**
		338 $a microopaque	if 007/01 is **g**
		338 $a other	if 007/01 is **u** or **z** or **|** [fill character]
		338 $a microform	if no 007 exists or no 007/01 is listed above
	Microscope slide	336 $a still image	
		337 $a microscope	
		338 $a microscope slide	
	Model	336 $a three-dimensional form	
		337 $a unmediated	

MARC	Current	New	Remarks
		338 $a object	
	Motion picture	336 $a two-dimensional moving image	
		337 $a projected	
		338 $a film cartridge	if 007/01 is **c**
		338 $a film cassette	if 007/01 is **f**
		338 $a film reel	if 007/01 is **o**
		338 $a film roll	if 007/01 is **r**
		338 $a other	if 007/01 is **u** or **z** or **|** [fill character]
		338 $a motion picture	if no 007 exists or no 007/01 is listed above
	Music	336 $a notated music	if LDR 06 is **c** or **d**
		337 $a unmediated	
		338 $a volume	
	Picture	336 $a still image	
		337 $a unmediated	
		338 $a object	
	Realia	336 $a three-dimensional form	
		337 $a unmediated	
		338 $a object	
	Slide	336 $a still image	
		337 $a projected	
		338 $a slide	
	Sound recording	336 $a performed music	if LDR 06 is **j**
		336 $a spoken word	if LDR 06 is **i**
		337 $a audio	
		338 $a audio cartridge	if 007/01 is **g**
		338 $a audio cylinder	if 007/01 is **e**
		338 $a audio disc	if 007/01 is **d**
		338 $a audio roll	if 007/01 is **q**
		338 $a audiocassette	if 007/01 is **s**
		338 $a audiotape reel	if 007/01 is **t**
		338 $a sound-track reel	if 007/01 is **i**
		338 $a other	if 007/01 is **u** or **w** or **z** or **|** [fill character]

MARC	Current	New	Remarks
		338 $a sound recording	if no 007 exists or no 007/01 is listed above
	Technical drawing	336 $a still image	
		337 $a unmediated	
		338 $a sheet	
	Text	336 $a text	
		337 $a unmediated	
		338 $a volume	
	Toy	336 $a three-dimensional form	
		337 $a unmediated	
		338 $a object	
	Transparency	336 $a still image	
		337 $a projected	
		338 $a slide	
	Video recording	336 $a two-dimensional moving image	
		337 $a projected	
		338 $a video cartridge	if 007/01 is **c**
		338 $a videocassette	if 007/01 is **f**
		338 $a videodisc	if 007/01 is **d**
		338 $a videotape reel	if 007/01 is **r**
		338 $a other	if 007/01 is **u** or **z** or **|** [fill character]
		338 $a video recording	if no 007 exists or no 007/01 is listed above
	Game	336 $a three-dimensional form	
		337 $a unmediated	
		338 $a object	
	Kit	336 $a three-dimensional form	
		337 $a unmediated	
		338 $a object	

17.1.2 Authority Records

(P = permanent, b = provisional or in bib headings)

MARC	Current	New	Remarks
100, 130, 500, 530 P or 100, 130, 600, 630, 700, 730, 800, 830 b	$a Bible $p O.T.	Old Testament $p O.T. is deleted	if no addit'l $p exists; e.g., *130 \a Bible \p O.T.* becomes *130 \a Bible \p Old Testament* if additional $p exists; e.g., *130 0 \a Bible. \p O.T. \p Ezra.* becomes *130 0 \a Bible. \p Ezra.*
	$a Bible $p N.T.	New Testament $p N.T. is deleted	if no addit'l $p exists; e.g., *130 0 \a Bible. \p N.T. \x Criticism, interpretation,* becomes *130 0 \a Bible. \p New Testament \x Criticism, interpretation,* if additional $p exists; e.g., *130 0 \a Bible. \p N.T. \p Acts* becomes *130 0 \a Bible. \p Acts*
	$d b.	(Hyphen follows birth date)	b. 1945 becomes 1945-
	$d d.	(Hyphen precedes all other data in $d)	d. 1945 becomes -1945

17.1.3 Permanent Authority Records

Note: If any change as described below, including any change to an abbreviation, is made to the permanent authority record, **040$e rdax** is added to the record.

MARC	Current	New	Remarks
4xx			NOT modified
1xx		4xx	Any modified 1xx should become a 4xx
008/10	Any value	z	If modified

17.2 Mapping of AACR2 Abbreviations to Terms in RDA

Abbreviation in AACR2	Term in RDA	Stored in: MARC Bibliographic Tag	MARC Authority Tag	Language, if other than English
abgedr.	Abgedruckt	245 $b, 250 $a, 500 $a		
Abt.	Abteilung	245 $c, 260 $b, 440 $a, 490 $v	130 $a, 530 $a	
acc.	accompaniment	500 $a		
afd.	afdeling	260 $b, 490 $a		
afl.	aflevering	440 $v, 490 $a		
&	and	730 $l	130 $l, 530 $l	
et al.	and others	245 $c		
A.D.	Anno Domini	100 $d, 500 $a, 600 $d, 650 $y, 651 $y, 700 $d	150 $y, 151 $y, 500 $d, 550 $y, 551 $y	
approx.	approximately	020 $c, 255 $a		
årg.	årgang	050 $b		Danish
arg.	argraffiad	250 $a		Welch
arr.	arranged	100 $o, 240 $o, 245 $c, 700 $o, 730 $o	100 $o, 130 $o, 500 $o, 530 $o	
átdolg.	átdolgozott	250 $a $b		
Aufl.	Auflage	250 $a $b		
augm.	augmenté	250 $a $b		French
augm.	augmented	250 $a $b		
aum.	aumentada	250 $a $b		Spanish
aum.	aumentato	250 $a $b		Italian
Ausg.	Ausgabe	250 $a $b 440 $a 500 $a	130 $a 530 $a	
avd.	avdeling	110 $b, 240 $l, 260 $b, 410 $t, 440 $a, 490 $a, 810 $t	110 $b, 110 $t, 130 $a, 510 $b, 510 $t, 530 $a	

Abbreviation in AACR2	Term in RDA	Stored in: MARC Bibliographic Tag	MARC Authority Tag	Language, if other than English
Bd.	Band	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		
bd.	band	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		
Bdchn.	Bändchen	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		
bdchn.	bändchen	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		
Bde.	Bände	020, $a, 245 $b, 250 $a $b, 490 $v, 500 $a, 505 $a		
B.C.	Before Christ	100 $d, 400 $d, 600 $d, 651 $y, 700 $d, 800 $d,	100 $d, 151 $y, 500 $d, 551 $y	
bil.	bilangan	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		
b&w	black and white	035 $c, 300 $b		
bogtr.	bogtrykkeri	260 $b $f		
boktr.	boktrykkeri	260 $b $f		
bk.	book	020 $a, 505 $a, 440 $v, 800 $v, 810 $v, 811 $v, 830 $v		
b.	born	100 $d, 600 $d, 700 $d, 800 $d	100 $d, 500 $d	
bőv.	bővített	250 $a,		
br.	broj	440 $v, 490 $v, 800 $v, 810 $v, 811 $v, 830 $v		
Bro.	Brother	110 $a, 111 $a, 610 $a, 611 $a, 710 $a, 711 $a, 810 $a, 811 $a	110 $a, 111 $a, 510 $a, 511 $a	
Bros.	Brothers	110 $a, 111 $a, 610 $a, 611 $a, 710 $a, 711 $a, 810 $a, 811 $a	110 $a, 111 $a, 510 $a, 511 $a	
Buchdr.	Buchdrucker	260 $b		
Buchh.	Buchhandlung	260 $b		
bull.	bulletin	210 $a		
bpi	bytes per inch	300 $a	100 $d, 500 $d	
cap.	capitolo	245 $a $p, 505 $a		Italian
č.	část	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		Czech
cent.	century	500 $a		

Abbreviation in AACR2	Term in RDA	Stored in: MARC Bibliographic Tag	MARC Authority Tag	Language, if other than English
cet.	cetakan	250 $a $b		
ch.	chapter	500 $a		
ca.	approximately	100 $d, 300 $a, 600 $d, 700 $d, 800 $d	100 $d, 500 $d	
čís.	číslo	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		
col.	colored	300 $b		
Cia.	Compagnia	260 $b, 500 $a		Italian
Cie	Compagnie	260 $b, 500 $a		French
Cía.	Compañía	260 $b, 500 $a		Spanish
Co.	Company	260 $b, 500 $a		
comp.	compiler	100 $e, 700 $e	100 $e, 500$e	
Corp.	Corporation	260 $b, 500 $a		
corr.	corrected	250 $a		
corr.	corregido	250 $a		Spanish
corr.	corretto	250 $a		Italian
corr.	corrigé	250 $a		French
cz.	część	505 $a		Polish
decl.	declination	255 $a		
d.	deel	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		Dutch
d.	del	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		Danish
d.	del	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		Norwegian
d.	del	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		Swedish
dép.	département	110 $b, 260 $b, 651 $a, 710 $b	110 $b, 151 $a, 510 $b, 551 $a	French
Dept.	Department	037 $b, 110 $b, 260 $b, 490 $b, 651 $a	110 $b, 151 $a, 510 $b, 551 $a	
diam.	diameter	300 $a $c		
doc.	document	440 $a, 500 $a	130 $a, 530 $a	

Abbreviation in AACR2	Term in RDA	Stored in: MARC Bibliographic Tag	MARC Authority Tag	Language, if other than English
dop.	dopunjeno	250 $a		Croatian
druk.	drukarnia	260 $a		Polish
ed.	edición	250 $a		Spanish
ed.	edition	250 $a		
eds.	editions	250 $a		
éd.	édition	250 $a		French
ed.	editor	100 $e, 700 $e, 800 $e	100 $e, 500$e	
ed.	edizione	250 $a		Italian
enl.	enlarged	250 $a		
eq.	equinox	255 $a		
erg.	ergänzt	250 $a		German
erw.	erweitert	250 $a		German
estab. tip.	establecimiento tipográfico	260 $b, 500 $a		Spanish
évf.	évfolyam	440 $v, 490 $v, 800 $v, 810 $v, 811 $v, 830 $v		
facsim.	facsimile	300 $b		
facsims.	facsimiles	300 $b		
fasc.	fascicle	440 $v, 490 $a		
fasc.	fascicule	440 $v, 490 $a		French
fl.	active	100 $d, 600 $d, 700 $d, 800 $d	100 $d, 500 $d	
ft.	foot, feet	255 $a, 300 $a		
fps	frames per second	500 $a		
f.lli	fratelli	110 $a $b, 410 $a $b, 610 $a $b, 710 $a $b, 810 $a $b	110 $a $b, 510 $a $b	
Gebr.	Gebrüder	110 $a $b, 410 $a $b, 610 $a $b, 710 $a $b, 810 $a $b	110 $a $b, 510 $a $b	
geneal.	genealogical	300 $b		
g.	godina	440 $a $v	130 $a, 530 $a	

Abbreviation in AACR2	Term in RDA	Stored in: MARC Bibliographic Tag	MARC Authority Tag	Language, if other than English
govt.	government	260 $b		
G.P.O.	Government Printing Office	260 $b		
Hs.	Handschrift	260 $b, 500 $a		
Hss.	Handschriften	260 $b, 500 $a		
H.M.S.O.	Her Majesty's Stationery Office	260 $b		
Hnos.	Hermanos	110 $a $b, 410 $a $b, 610 $a $b, 710 $a $b, 810 $a $b	110 $a $b, 510 $a $b	
ill.	illustrations	300 $b		
ill.	illustrator	100 $e, 700 $e, 800 $e	100 $e, 500$e	
in.	inches	300 $b		
ips	inches per second	300 $b		
incl.	including	300 $b		
introd.	introduction	245 $c		
km.	kilometres	500 $a, 507 $b		
ms.	manuscript	245 $a $c, 500 $a		
mss.	manuscripts	245 $a $c, 500 $a		
min.	minutes	300 $b		
mono.	monophonic	300 $b		
new ser.	new series	440 $v, 490 $v, 800 $v, 810 $v, 811 $v, 830 $v		
no.	nombor	440 $v, 490 $v, 800 $v, 810 $v, 811 $v, 830 $v		Malay
no.	nomor	440 $v, 490 $v, 800 $v, 810 $v, 811 $v, 830 $v		Indonesian
no.	number	440 $v, 490 $v, 800 $v, 810 $v, 811 $v, 830 $v		

Abbreviation in AACR2	Term in RDA	Stored in: MARC Bibliographic Tag	MARC Authority Tag	Language, if other than English
n:o	numero	440 $v, 490 $v, 800 $v, 810 $v, 811 $v, 830 $v		Finnish
no	numéro	440 $v, 490 $v, 800 $v, 810 $v, 811 $v, 830 $v		French
n.	numero	440 $v, 490 $v, 800 $v, 810 $v, 811 $v, 830 $v		Italian
no.	número	440 $v, 490 $v, 800 $v, 810 $v, 811 $v, 830 $v		Spanish
nr.	numer	440 $v, 490 $v, 800 $v, 810 $v, 811 $v, 830 $v		Polish
Nr.	Nummer	440 $v, 490 $v, 800 $v, 810 $v, 811 $v, 830 $v		German
nr.	nummer	440 $v, 490 $v, 800 $v, 810 $v, 811 $v, 830 $v		German
op.	opus	240 $n, 245 $a $b, $246 $a, 700 $n	100 $n, 500 $n	
p.	pages	300 $a, 500 $a, 504 $a		
pbk.	paperback	020 $a		
pt.	part	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		
pts.	parts	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		
pt.	parte	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		Spanish
ptie	partie	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		French
pties	parties	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		French
photo.	photograph	300 $b		
photos.	photographs	300 $b		
popr.	poprawione	250 $a		
port.	portrait	300 $b		
ports.	portraits	300 $b		
pref.	preface	245 $a $c, 500 $a		
prelim.	preliminary	500 $a, 504 $a		
print.	printing	500 $a		
proj.	projection	255 $b, 800 $c	100 $c, 500 $c	

Abbreviation in AACR2	Term in RDA	Stored in: MARC Bibliographic Tag	MARC Authority Tag	Language, if other than English
pseud.	pseudonym	100 $c, 400 $c, 500 $a, 600 $a, 700 $c	100 $a $c, 500 $a $c	
pub.	publishing	260 $b		
quad.	quadraphonic	300 $b		
rept.	report	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		
repr.	reprinted	250 $a		
reprod.	reproduced	245 $c,260 $b, 500 $a		
rev.	revisé	250 $a		Spanish
rev.	revised	250 $a		
rpm	revolutions per minute	300 $a $b $e		
rev.	revu	250 $a		French
sec.	seconds	300 $a $b $e		
sér.	série	440 $v, 490 $v, 800 $v, 810 $v, 811 $v, 830 $v		French
ser.	series	440 $v, 490 $v, 800 $v, 810 $v, 811 $v, 830 $v		
si.	silent	300 $b		
s.l.	place of publication not idenified	260 $a		
s.n.	publisher not identified	260 $b		
sd.	sound	300 $b		
stereo.	stereophonic	300 $b		
Supt. of Docs.	Superintendent of Documents	260 $b		
suppl.	supplement	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		

Abbreviation in AACR2	Term in RDA	Stored in: MARC Bibliographic Tag	MARC Authority Tag	Language, if other than English
T.	Teil	440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		German
t.p.	title page	500 $a		
tr.	translator	100 $e, 700 $e, 800 $e	100 $e, 500 $e	
unacc.	unaccompanied	500 $a		
v.	volume	300 $a, 440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		
vol.	volume	300 $a, 440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		
vols.	volumes	300 $a, 440 $v, 490 $a, 505 $a, 800 $v, 810 $v, 811 $v, 830 $v		
Jan.	January	100 $d, 110 $d, 111 $d, 400 $d, 410 $d, 411 $d, 600 $d, 610 $d, 611 $d, 700 $d, 710 $d, 711 $d, 800 $d, 810 $d, 811 $d	100 $d, 110 $d, 111 $d, 500 $d, 510 $d, 511 $d	
Feb.	February	100 $d, 110 $d, 111 $d, 400 $d, 410 $d, 411 $d, 600 $d, 610 $d, 611 $d, 700 $d, 710 $d, 711 $d, 800 $d, 810 $d, 811 $d	100 $d, 110 $d, 111 $d, 500 $d, 510 $d, 511 $d	
Mar.	March	100 $d, 110 $d, 111 $d, 400 $d, 410 $d, 411 $d, 600 $d, 610 $d, 611 $d, 700 $d, 710 $d, 711 $d, 800 $d, 810 $d, 811 $d	100 $d, 110 $d, 111 $d, 500 $d, 510 $d, 511 $d	
Apr.	April	100 $d, 110 $d, 111 $d, 400 $d, 410 $d, 411 $d, 600 $d, 610 $d, 611 $d, 700 $d, 710 $d, 711 $d, 800 $d, 810 $d, 811 $d	100 $d, 110 $d, 111 $d, 500 $d, 510 $d, 511 $d	
Jun.	June	100 $d, 110 $d, 111 $d, 400 $d, 410 $d, 411 $d, 600 $d, 610 $d, 611 $d, 700 $d, 710 $d, 711 $d, 800 $d, 810 $d, 811 $d	100 $d, 110 $d, 111 $d, 500 $d, 510 $d, 511 $d	
Jul.	July	100 $d, 110 $d, 111 $d, 400 $d, 410 $d, 411 $d, 600 $d, 610 $d, 611 $d, 700 $d, 710 $d, 711 $d, 800 $d, 810 $d, 811 $d	100 $d, 110 $d, 111 $d, 500 $d, 510 $d, 511 $d	
Aug.	August	100 $d, 110 $d, 111 $d, 400 $d, 410 $d, 411 $d, 600 $d, 610 $d, 611 $d, 700 $d, 710 $d, 711 $d, 800 $d, 810 $d, 811 $d	100 $d, 110 $d, 111 $d, 500 $d, 510 $d, 511 $d	
Sept.	September	100 $d, 110 $d, 111 $d, 400 $d, 410 $d, 411 $d, 600 $d, 610 $d, 611 $d, 700 $d, 710 $d, 711 $d, 800 $d, 810 $d, 811 $d	100 $d, 110 $d, 111 $d, 500 $d, 510 $d, 511 $d	
Oct.	October	100 $d, 110 $d, 111 $d, 400 $d, 410 $d, 411 $d, 600 $d, 610 $d, 611 $d, 700 $d, 710 $d, 711 $d, 800 $d, 810 $d, 811 $d	100 $d, 110 $d, 111 $d, 500 $d, 510 $d, 511 $d	

Abbreviation in AACR2	Term in RDA	Stored in: MARC Bibliographic Tag	MARC Authority Tag	Language, if other than English
Nov.	November	100 $d, 110 $d, 111 $d, 400 $d, 410 $d, 411 $d, 600 $d, 610 $d, 611 $d, 700 $d, 710 $d, 711 $d, 800 $d, 810 $d, 811 $d	100 $d, 110 $d, 111 $d, 500 $d, 510 $d, 511 $d	
Dec.	December	100 $d, 110 $d, 111 $d, 400 $d, 410 $d, 411 $d, 600 $d, 610 $d, 611 $d, 700 $d, 710 $d, 711 $d, 800 $d, 810 $d, 811 $d	100 $d, 110 $d, 111 $d, 500 $d, 510 $d, 511 $d	

18. Appendix D – Permalinks Scripts and Executables

Libraries that wish to use Innovative's permalinks functionality must first add permalink 024 tags to the bibliographic and authority records in their database(s). The sections below describe the following programs and scripts that can be used to add permalink 024 tags to bibliographic and authority records:

- **SetPermalinkUI.sh** - Determines the website used as the base of the permalinks in your collection.
- **SetAutoAddPermalinksFlag.sh** - Running this script configures Virtua to automatically create a permalink 024 tag when a bibliographic or authority record is cataloged.
- **Create024PermalinkTagInBibRecords.exe** - This program adds permalink 024 tags to existing bibliographic records.
- **Create024PermalinkTagInAuthorityRecords.exe** - This program adds permalink 024 tags to existing authority records.
- **Index24PermalinkTagInAuthorityRecords.exe** - This program indexes permalink 024 tags in authority records.

Note: A **permalink** is a permanent web address (i.e., a URL) that points to specific content. A permalink remains unchanged indefinitely and continues to direct users to the specific content even if the physical location of the specific content is changed over time.

Innovative's permalink functionality allows libraries to create permanent links to bibliographic and authority record content stored in a library's database (or databases). For information on purchasing this feature, contact Innovative Customer Services.

18.1 Determining the URI for Permalinks

Purpose: Determines the universal resource identifier (usually a website) used as the base of the permalinks in your collection.

Usage: `SetPermalinkUI.sh <permalinkURI>`

Where **<permalinkURI>** is the URI that makes up the base of the permalink; for example:

```
SetPermalinkUI.sh http://permalinks.vtlslibrary.com/
```

18.2 Enabling Automatic Creation of Permalink 024 Tags

Purpose: Configures Virtua to automatically create permalink 024 tags in bibliographic and authority records when a record is cataloged.

Usage: `SetAutoAddPermalinksFlag.sh <value>`

Where **<value>**:

0 - Do not add permalink 024 tags to bibliographic and authority records.
1 - Add permalink 024 tags to bibliographic and authority records.

If you wish to use permalinks, run this script once with the value set to 1 (add permalink 024 tags). Permalink 024 tags will be automatically added to all bibliographic and authority records cataloged after the script is run.

18.3 Adding Permalink 024 Tags to Existing Bibliographic Records

Purpose: Adds permalink 024 tags to existing bibliographic records.

Usage: `Create024PermalinkTagInBibRecords.exe < fileOfBibIds1 > fileOfBibIds2`

Where:

fileOfBibIds1 = A file of bib-IDs to add the 024 tag to.
fileOfBibIds2 = A file of bib-IDs to which the 024 tag was added.

If you wish to use permalinks, run this program once to add permalink 024 tags to existing bibliographic records.

Note: After permalink 024 tags are added to the bibliographic records, you must re-index bibliographic record data in Chamo once to add the 024 tag to the Solr index. Future indexing will be handled by the cataloging module.

18.4 Adding Permalink 024 Tags to Existing Authority Records

Purpose: Adds permalink 024 tags to existing authority records.

Usage: `Create024PermalinkTagInAuthorityRecords.exe < fileOfAuthIds1 > fileOfAuthIds2`

Where:

`fileOfAuthIds1` = A file of auth-IDs to add the 024 tag to.
`fileOfAuthIds2` = A file of auth-IDs to which the 024 tag was added.

If you wish to use permalinks, run this program once to add permalink 024 tags to existing authority records.

18.5 Indexing Authority Records After Adding Permalink 024 Tags

After permalink 024 tags are added to the authority records, you must run the program **Index24PermalinkTagInAuthorityRecords.exe** once to index data. Future indexing will be handled by Virtua's cataloging module.

Purpose: Index permalink 024 tags in authority records.

Usage: `Index24PermalinkTagInAuthorityRecords.exe < fileOfAuthIds >`

Where:

`fileOfAuthIds` = A file of auth-IDs of authority records.

19. Appendix E - Changes in this Guide

19.1 Changes for 16.1

No changes were made.

19.2 Changes for Version 15.2

"Deleting Items by Location" - The **DeleteItemsByLocation.ksh** script was updated to change the filename extension from "**.ksh**" to "**.sh**". The new script functions exactly as the old one did.

Index

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

ß

S

T

U

V

W

X

Z