User's Guide

Integrated Library System

System Management: Circulation

VIRTUA ILS – INTEGRATED LIBRARY SYSTEM

Virtua System Management

Circulation 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 58

1. Introduction

In this user's guide, we describe the procedures for running executables and scripts related to the Circulation subsystem of the Virtua ILS – Integrated Library System. Information in this guide is intended for system administrators who feel comfortable working with programs that directly modify the contents of the Virtua database.

Note: From this point on, we will refer to the Virtua ILS – Integrated Library System as simply Virtua or the Virtua system in this guide.

Most of the scripts and executables that we describe in this guide are not programs that you need to run daily. If you are not sure 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 script or executable, or if you are unaware of the consequences of running a script or executable, contact Innovative before 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 good backups of your database. For information about database backups, see the *System Management Reference Guide*.
- 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.

Throughout this guide, we assume that you have already read the *Virtua System Management Reference Guide*, which provides an introduction to the Virtua system, administrative tips for working with UNIX and Oracle, information on environment variables, and various general information about system administration.

1.1 Using this Guide

The *System Management: Circulation User's Guide* contains instructions for running the scripts and executables related to the Circulation subsystem. You can use the list below and the Table of Contents to locate specific information in this guide.

For . . .	See . . .
Information about diagnosing and troubleshooting problems with circulation parameters	Chapter 2
Instructions for running a program that deletes patrons whose records have expired or are inactive	Chapter 3
Instructions for running a program that sets the value of tag 015 subfield $b of a batch of patron records.	Chapter 4
Instructions for deleting unencrypted patron passwords from the patron records in your database.	Chapter 5
Instructions for extracting a list of IDs for the patron records in your database	Chapter 6
Instructions for resetting to zero the temporary circulation count of a set of specified item records	Chapter 7
Instructions for running a script that deletes entries from the transaction log	Chapter 8
Instructions for running a script that removes identifying patron information from the transaction log	Chapter 9
Instructions for converting circulation related currency values	Chapter 10
Instructions for extracting barcodes of checked out items	Chapter 11
Instructions for calculating fines for overdue items that have not yet been checked in	Chapter 12
Reference information about the scripts and executables discussed in this guide	Appendix A
A list of the transaction type codes used to store information in the CIRC_TRANSACTION_LOG database table	Appendix B
A list of changes in this document since the last version	Appendix C

Note:
- Scripts and executables for the Circulation Backup System are documented in the *Circulation Backup System User's Guide*.
- Scripts and executables for the 3M SelfCheck Interface are documented in the *3M Patron SelfCheck Interface Startup Guide*.

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. Checking Circulation Parameters

The script **CheckCircParameters.sh** evaluates the circulation parameters in your database and returns a list of errors or warnings regarding those parameters. This script checks for missing parameters and identifies them based on the item classes, patron types, and locations that your library has defined. It also examines the items and patrons in the database and reports on any locations, item classes, and patron types that are in use but not defined.

You can fix most of the problems reported by this script by modifying the affected record or by setting a parameter in the Virtua Profiler. In the section "Troubleshooting Problems with Circulation Parameters," we suggest a solution for each problem that this script reports.

This chapter covers the following topics:

2.1 Running CheckCircParameters.sh

To check for problems with circulation parameters,

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

2. At the prompt, type: **`CheckCircParameters.sh`**

   The script runs and checks the circulation parameters in your database and reports any problems in a file named **CheckCircParameters.log**

3. View the problems reported in **CheckCircParameters.log**. For additional information, see the following section.

2.2 Troubleshooting Problems with Circulation Parameters

The script **CheckCircParameters.sh** writes error and warning messages to a file called **CheckCircParameters.log**. The following sections list, describe, and provide a possible solution for each problem that the script can report.

2.2.1 Error: The following owning locations are not defined . . .

Description: The script found one or more item records that specify an owning location that does not exist. The script lists the location code for each location that it could not find.

Solution: Find the item records that specify the non-existent locations, and, in the Virtua client, assign an existing owning location. Below, we provide an SQL query for finding the item ID of all records with a specified owning location:

select itemid from itemdetl2 where location = '[location code]'

Where **[location code]** is a location code listed by the script.

In the Virtua client, find each item ID returned by the query, and assign the associated item an existing owning location. For information about modifying item records, see the *Virtua Cataloging User's Guide*.

2.2.2 Error: The following shelf locations are not defined . . .

Description: The script found one or more item records that specify a shelving location that does not exist. The script lists the location code for each location that it could not find.

Solution: Find the item records that specify the non-existent locations, and, in the Virtua client, assign an existing shelving location. Below, we provide an SQL query for finding the item ID of all records with a specified shelving location:

select itemid from itemdetl2 where shelflocation = '[location code]'

Where **[location code]** is a location code listed by the script.

In the Virtua client, find each item ID returned by the query, and assign the associated item an existing shelving location. For information about modifying item records, see the *Virtua Cataloging User's Guide*.

2.2.3 Error: The following item classes are not defined . . .

Description: The script found one or more item records that specify an item class code that was not defined in the Item Class Definitions parameter in the Virtua Profiler. The script lists each item class code that it could not find.

Solution: Find the item records that specify the non-existent item class code, and, in the Virtua client, assign an existing item class. Below, we provide an SQL query for finding the item ID of all records with a specified item class code:

select itemid from itemdetl2 where itemclass = '[item class code]'

Where **[item class code]** is an item class code listed by the script.

In the Virtua client, find each item ID returned by the query, and assign the associated item an existing item class. For information about modifying item records, see the *Virtua Cataloging User's Guide*.

2.2.4 Error: The following reserve item classes (on reserve) are not defined . . .

Description: The script found one or more item records that specify a reserve item class code that was not defined as a reserve item class in the Item Class Definitions parameter in the Virtua Profiler. The script lists each item class code that it could not find.

Solution: Find the item records that specify the non-existent reserve item class code, and, in the Virtua client, assign an existing reserve item class. Below, we provide an SQL query for finding the item ID of all records with a specified item class code:

select itemid from itemdetl2 where tempitemclass1 = '[item class code]'

Where **[item class code]** is an item class code listed by the script.

In the Virtua client, find each item ID returned by the query, and assign the associated item an existing reserve item class. For information about modifying item records, see the *Virtua Cataloging User's Guide*.

2.2.5 Error: The following patron types are not defined . . .

Description: The script found one or more patron records that specify a patron type code in subfield $a of the 030 tag that is not defined in the Patron Types parameter. The script lists each patron type code that it could not find.

Solution: Find the patron records that specify the non-existent patron type, and change the value in subfield $a of 030 tag to a valid patron type code. Below, we provide an SQL query for finding the patron barcode of all records with a specified patron type:

select barcode from patron where patron_type_id='[patron type code]'

Where **[patron type code]** is a patron type code listed by the script.

In the Virtua client, find each barcode returned by the query, and assign the associated patron record an existing patron type code. For information about modifying patron records, see the *Circulation Control/Patron Information User's Guide*.

2.2.6 Error: The following patron types (used in requests) are not defined . . .

Description: This message does not normally indicate a serious problem beyond what is reported by the *Error: The following patron types are not defined . . .* message. In version 43.0 of this software, this message will not appear.

2.2.7 Error: location . . . has no location values set

Description: The script found a location that has no associated values set in the Location Names parameter.

Solution: Open the associated Location Names parameter for the specified location in the Virtua Profiler. Set the parameters as desired, and save the changes.

For information about working with the Location Names parameter, see the *Virtua Profiler/Global Settings User's Guide*.

2.2.8 Info: location . . . is not a member of any request group

Description: The script found a location that is not a member of a request group. This message will be returned only when one or more of your Global Request Trapping Options is set to Group.

Solution: Add the specified location to a request group in the Virtua Profiler.

For information about request groups, see the *Virtua Profiler/Circulation Parameters User's Guide*.

2.2.9 Info: location . . . has no closed date(s) set

Description: The script found a location for which there are no closed dates set in the Closed Dates parameter. This message may not indicate a critical problem.

Solution: If desired, add one or more closed dates for the specified location in the Closed Dates parameter of the Virtua Profiler.

For information about adding closed dates, see the *Virtua Profiler/Circulation Parameters User's Guide*.

2.2.10 Error: location . . . has no library hours set

Description: The script found a location for which there are no library hours set.

Solution: Configure the Library Hours parameter for the specified location in the Virtua Profiler.

For information about configuring the Library Hours parameter, see the *Virtua Profiler/Circulation Parameters User's Guide*.

2.2.11 Error: location . . . has no library-defined due date(s)

Description: The script found a location for which there are no library-defined due dates set. This message does not appear unless the Loan Rules set for the Location + Patron + Item Matrix specifies a fixed date.

Solution: There are two solutions to this problem:

- For the specified location, configure the Library Defined Dates parameter in the Virtua Profiler.
  -OR-
- On the Loan Rules tab of the Location + Patron + Item matrices associated with specified location, choose Calculated loan periods rather than Fixed loan periods.

For information about configuring the Library Defined Dates parameter or the Location + Patron + Item Matrix, see the *Virtua Profiler/Circulation Parameters User's Guide*.

2.2.12 Error: alerts and blocks are not completely defined for patron type . . .

Description: The script found a patron type for which the Alerts and Blocks Matrix is incomplete or missing.

Solution: For the patron type specified, configure the Alerts and Blocks Matrix in the Virtua Profiler.

For information about configuring the Alerts and Blocks parameter, see the *Virtua Profiler/Circulation Parameters User's Guide*.

2.2.13 Error: no checkout limits defined for patron type . . .

Description: The script found a patron type for which check-out limits have not been defined.

Solution: For the patron type specified, configure the Check-out Limits Matrix in the Virtua Profiler.

For information about configuring the Check-out Limits Matrix, see the *Virtua Profiler/Circulation Parameters User's Guide*.

Note: This error message will not appear unless the Check-out Limits Matrix field on the Check-out Limits Type window is set to Patron Type.

2.2.14 Error: location+patron checkout limits missing for . . .

Description: The script found a location and patron type combination for which check-out limits have not been defined.

Solution: For the location and patron type specified, configure the Check-out Limits Matrix in the Virtua Profiler.

For information about configuring the Check-out Limits Matrix, see the *Virtua Profiler/Circulation Parameters User's Guide*.

Note: This error message will not appear unless the Check-out Limits Matrix field on the Check-out Limits Type window is set to Patron Type + Location.

2.2.15 Error: patron+item checkout limits missing for . . .

Description: The script found a patron type and item class combination for which check-out limits have not been defined.

Solution: For the patron type and item class specified, configure the Check-out Limits Matrix in the Virtua Profiler.

For information about configuring the Check-out Limits Matrix, see the *Virtua Profiler/Circulation Parameters User's Guide*.

Note: This error message will not appear unless the Check-out Limits Matrix field on the Check-out Limits Type window is set to Patron Type + Item.

2.2.16 Error: location+patron+item checkout limits missing for . . .

Description: The script found a location, patron type, and item class combination for which check-out limits have not been defined.

Solution: For the location, patron type, and item class specified, configure the Check-out Limits Matrix in the Virtua Profiler.

For information about configuring the Check-out Limits Matrix, see the *Virtua Profiler/Circulation Parameters User's Guide*.

Note: This error message will not appear unless the Check-out Limits Matrix field on the Check-out Limits Type window is set to Patron Type + Item + Location.

2.2.17 Error: location+patron matrix missing for . . .

Description: The script found a location and patron type combination in the Location + Patron Matrix for which no parameters are defined.

Solution: For the location and patron type specified, configure the Location + Patron Matrix in the Virtua Profiler. Even though there are no parameters defined in the matrix, the window for the matrix does exist in the Profiler. When you open this matrix window, the text in the lower left corner will read *NOT SET!*

For information about circulation matrices, see the *Virtua Profiler/Circulation Parameters User's Guide*.

2.2.18 Error: location+item matrix missing for . . .

Description: The script found a location and item type combination in the Location + Item Matrix for which no parameters are defined.

Solution: For the location and item class specified, configure the Location + Item Matrix in the Virtua Profiler. Even though there are no parameters defined in the matrix, the window for the matrix does exist in the Profiler. When you open this matrix window, the text in the lower left corner will read *NOT SET!*

For information about circulation matrices, see the *Virtua Profiler/Circulation Parameters User's Guide*.

2.2.19 Error: location+patron+item matrix missing for . . .

Description: The script found a location and patron type combination in the Location + Patron + Item Matrix for which no parameters are defined.

Solution: For the location, patron type, and item class specified, configure the Location + Patron + Item Matrix in the Virtua Profiler. Even though there are no parameters defined in the matrix, the window for the matrix does exist in the Profiler. When you open this matrix window, the text in the lower left corner will read *NOT SET!*

For information about circulation matrices, see the *Virtua Profiler/Circulation Parameters User's Guide*.

3. Deleting Patron Records

There are four scripts that can be used to delete patron records from your database. Each of three scripts, described below, has a unique purpose and uses **vload.exe** to delete the records. The use of **vload.exe** means that a patron record will not be deleted if any outstanding circulation activity, fines/fees, requests, etc. is associated with the record. The three scripts that use **vload.exe** are:

- **DeleteExpiredPatrons.sh** – Generates a list of patron IDs and deletes the patron records (unless otherwise specified) based on the expiration date of the patron records.

- **DeleteInactivePatrons.sh** – Generates a list of patron IDs and deletes the patron records (unless otherwise specified) based on one of the following two criteria:
  - Last activity date of the patron record.
  - Length of time (in months or years) that the patron record has been inactive.

- **RemovePatronsByDeletionDate.sh** – Generates a list of patron IDs and deletes the patron records (unless otherwise specified) whose deletion date has passed. The deletion date is stored in the 042 tag subfield $c of the patron record and is set at cataloging time based on the Patron Delete setting in the Location + Patron matrix in the Virtua Profiler.

A fourth script **DeletePatrons.sh** deletes patron records based on an input file of patron IDs *regardless* of a record's outstanding associations. The script first removes and logs any associated outstanding fines/fees, check-outs, requests, reserve instructor entries, and serial routing entries and then deletes the records. **DeletePatrons.sh** also outputs three files.

Important: Before you run any of the scripts that delete patron records, consider the following:

1. These programs *permanently* delete patron records from your database. Keep in mind that at any time before the program starts to execute your commands, you can press the CTRL + C keys to exit the script
2. If you do not have a recent export that you can use for recovery, we strongly recommend that you use **write2709.exe** to extract to a file the patron records in your database *before* you run these scripts. If you make a mistake when running these programs, you could use this file of records to recover to the previous state. For information on **write2709.exe**, see the *Virtua System Management: Cataloging User's Guide*.

3. If you mistakenly delete patron records and you do not have an export of the patron records that was taken before you ran these scripts, you will NOT be able to recover the deleted records.

This chapter covers the following topics:

3.1 Running DeleteExpiredPatrons.sh

To run DeleteExpiredPatrons.sh,

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

2. At the prompt, type:

```
DeleteExpiredPatrons.sh expirationDate(YYYYMMDD)
[only_print_file 1|0] [institution_symbol]
```

…where **[only_print_file 1|0]** allows you to specify via a "1" or "0" whether you want the script to generate a file of patron IDs only (but not delete records) that meet the criteria. To generate a file only, type **1**.

Note: Consortium databases must type the institution symbol of the institution from which patrons are to be deleted. If a patron belongs to more than one institution, the patron must be deleted from additional institutions before being deleted from the owning institution.

For example, to generate only a file of patron IDs whose records expired before January 1, 2005, type:

```
$EXE_DIR/DeleteExpiredPatrons.sh 20050101 1
```

3.2 Running DeleteInactivePatrons.sh

To run DeleteInactivePatrons.sh,

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

2. At the prompt, type:

```
DeleteInactivePatrons.sh [institution_symbol]
```

> **Note:** Consortium databases must type the institution symbol of the institution from which patrons are to be deleted. If a patron belongs to more than one institution, the patron must be deleted from additional institutions before being deleted from the owning institution.

3. Follow the prompts, as shown below, to specify the outcome you want and to input the criteria for deletion.

```
Would you like to generate ONLY a file of patrons IDs? No
patrons will be deleted.
(y/n)

y

Qualifying patron ids will be written to file: patron.ids
Please select one of the following methods to determine
eligible patrons:
  Enter 1 to specify the last activity date for the patron
records you wish to select.
  Enter 2 to specify the length of time (in months or years)
that the patron records have been inactive.

1

Please enter the last activity date for the patron records you
wish to select.
Example: To select all patrons that have not had any activity
since January 1, 2005, enter 20050101
Format: YYYYMMDD

20120101
```

3.3 Running RemovePatronsByDeletionDate.sh

To run RemovePatronsByDeletionDate.sh,

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

2. At the prompt, type:

   ```
   RemovePatronsByDeletionDate.sh [only_print_file 1|0]
   [institution_symbol]
   ```

   …where **[only_print_file 1|0]** allows you to specify via a "1" or "0" whether you want the script to generate a file of patron IDs only (but not delete records) that meet the criteria. To generate a file only, type **1**.

   Note: Consortium databases must type the institution symbol of the institution from which patrons are to be deleted. If a patron belongs to more than one institution, the patron must be deleted from additional institutions before being deleted from the owning institution.

   For example, to only generate a file of patron IDs whose records have a deletion date that has passed, type:

   ```
   $EXE_DIR/RemovePatronsByDeletionDate.sh 1
   ```

3.4 After Running a Delete Patrons Script…

Once any of the three scripts discuss above finishes running, **vload.exe** creates a log file named **vload_[time/date].log**, where **[time/date]** is the time and date at which the script called **vload.exe**. At the top of this log file, you can view a list of the records that the program failed to delete. Each failed record is listed in the following format:

<DIAGNOSTIC CODE *[code]*> for id = *[Patron ID]* idtype = 105>

Where . . .

- **[code]** is the diagnostic code for the error that occurred. For a list of diagnostic codes and their meanings, see the section "Diagnostic Codes" in this user's guide.
- **[Patron ID]** is the value of the 001 tag of the patron record that was not deleted.

Note: If you want to delete the patrons who were not deleted because of one of the reasons indicated by the diagnostic codes, add those patron IDs to a list that you will input to run **DeletePatrons.sh**, which deletes patron records regardless of outstanding associations.

3.5 Diagnostic Codes

In the table below, we list diagnostic codes that can appear for records that were not deleted by any one of the three scripts **DeleteInactivePatrons.sh**, **RemovePatronsByDeletionDate.sh**, **RemovePatronsByDeletionDate.sh**.

Diagnostic Code	Description
5401	The patron must return circulating items before the record can be deleted.
5402	Items reserved by the patron must be processed before deleting the record.
5403	The patron must pay outstanding fees before deleting the record, or the system outstanding-fee-waive flag must be ON.
5404	The patron must process outstanding requests before the record can be deleted.
5405	Cannot delete patron; MasterPatronDelete failed.
5406	Cannot delete patron. Failed to access item checkout information.
5407	Cannot delete patron. Failed to access item reserve information.
5408	Cannot delete patron. Failed to access outstanding fee information.
5409	Cannot delete patron. Failed to access outstanding request information.
5410	Cannot delete patron. Failed to access outstanding-fee-waive-flag.
5411	Cannot delete patron. Failed to access outstanding-request-deletion flag.
5412	Cannot delete patron. Failed to delete outstanding fee entries.
5413	Cannot delete patron. Failed to delete patron's outstanding request entries.

5414	Patron has waived fees and the reinstate fee right flag is active. Either reinstate and pay the fees, or delete the fees, or set the reinstate fee right flag OFF before deleting the patron.
5415	Cannot delete this patron because he/she is a member of a routing list. Remove patron's routing list entry before deleting patron record.
5416	Cannot delete patron record - serials routing list database error.
5417	Cannot delete patron record - patron is registered at more than one institution.
5418	Cannot delete patron record -- patron is not a member of your institution.
5419	Cannot delete patron record - patron has a current stack request or reservation.

3.6 Running DeletePatrons.sh

To run the script DeletePatrons.sh,

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

2. At the prompt, type:

`DeletePatrons.sh <filename of patron IDs>` [institution_symbol]

Where **[filename]** is the name of a file of patron IDs whose records you would like to delete. If you do not specify a filename, the script assumes that the filename is **patron.ids**.

Note: Consortium databases must type the institution symbol of the institution from which patrons are to be deleted. If a patron belongs to more than one institution, the patron must be deleted from additional institutions before being deleted from the owning institution.

DeletePatrons.sh removes and logs all outstanding elements associated with each patron record and then deletes each patron record whose ID is listed in the input file. The script outputs the following three files:

- **<filename>.items** - A file of item IDs of items that are checked out to deleted patrons.

- **<filename>.delinquent** - A file of patron IDs of patrons who are delinquent (i.e., where InfoStation has set their delinquent flag to 1), which may be used to send data to a collection agency.
- **<filename>.fees** - A file of patron IDs and the outstanding fees associated with the deleted patron records.

For items that were checked out to patrons whose records were deleted, Virtua will change their circulation state to "Available."

Note: DeletePatrons.sh deletes ALL of the patrons in the list of patron IDs, so *before* you run the script, be sure that you really wish to delete all of the patron records in your input file.

4. Updating Patron Passwords

The script **Update_Patron_015b.sh** adds or updates tag 015, subfield $b for the patron records specified by a file of patron IDs. Tag 015, subfield $b serves as a password field with tag 015, subfield $a serving as a username field. This script inserts the same value to tag 015, subfield $b for every record that it processes.

The script prompts you for responses. Thus per your direction, it either inserts a tag 015 $b or deletes an existing tag 015 $b and inserts a new one.

Before you run this script, you need to create a file that lists the IDs (001 tag) of the patron records that you want the script to process. The script prompts you as to the directory where the file will need to reside. That same directory will contain the log file generated from the script.

Tip: You can use the program **WritePatronIdsFile.ksh** to generate a list of record IDs for all patrons in your database. For information about this program, see the chapter "Extracting Patron IDs" in this user's guide.

4.1 Running Update_Patron_015b.sh

To add or update the 015 tag, subfield $b, for a batch of patron records,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type: **`Update_Patron_015b.sh`**

   The script displays a few lines of introductory comments.
3. Press Enter to continue.

   The script displays a list of things that will happen during processing.
4. Press Enter to continue.

   The script prompts you for the value that you want to assign to the 015 subfield $b tag.

5. Type the value that you want to assign to the subfield.

6. Press Enter.

   The script prompts you for the name and location of the file that contains the list of patron IDs.

7. Type the filename and edit the location if needed.

8. Press Enter.

   The script runs **ExportPatronBarcodeTable.sh** to export the contents of the ISO_2709 table. This table contains a large amount of patron data, so this process will take a long time. By exporting this table, the script allows you to return to a previous state if the results of the update are not what you expect.

   Once the script finishes exporting the table, it updates the patron records as you requested.

9. Check your patron records to make sure that the script made the desired changes. If there are problems with the changes that were made, you can recover to the previous state by using the export file that was generated by the script. This file is stored in the directory from which you ran the script. It has a prefix of **exp_iso_2709** and ends with the extension **.dat**.

5. Deleting Unencrypted Patron Passwords

All patron passwords are stored in encrypted format. As part of the migration process to Virtua version 48 or higher, passwords in subfield $b of the 015 and 016 tags of the patron record are used to generate new encrypted passwords. Encrypted passwords are stored in a new subfield $c of the record.

After a migration, you have the option of using the script **DeletePatronPasswords.sh** to delete unencrypted patron passwords in subfield $b of the 015 and 016 tags from all patron records that have the new encrypted version in subfield $c.

5.1 Running DeletePatronPasswords.sh

To delete all unencrypted patron passwords from the database,

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

All unencrypted patron passwords in subfield $b of the 015 and 016 tags are deleted from patron records that have the new encrypted version in subfield $c. Check your patron records to make sure that the script made the desired changes.

Note: Output from this script is automatically logged to the **DeletePatronPasswords.log** file.

6. Deleting Patron Fines and Fees

The script **DeleteOldFines.sh** lets you delete all patron fees and fines that have an assessment date that is earlier than a given date. After the script is run, it outputs a tab-delimited file of the following information about the deleted fines:

- Patron ID
- Amount
- Assessment date
- Item ID
- Fine Code
- State
- Bib ID
- Item class

The file is saved as **fines.deleted.YYYYMMDD**, where **YYYYMMDD** is the date chosen while running the script.

6.1 Running DeleteOldFines.sh

To delete patron fines and fees,

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

   The script then asks you to enter a date. The script deletes fines and fees assessed before this date.

   First the script asks you to enter a four-digit year.
4. Enter a four-digit year, and press Enter.

   The script asks you to enter a two-digit month
5. Enter a two-digit month, and press Enter.

The script asks you to enter a two-digit day.

6. Enter a two-digit day, and press Enter.

The fines and fees that were assessed before this date will be deleted, and the information is written to **fines.deleted.YYYYMMDD**, where YYYYMMDD is the date you entered when running the script.

7. Extracting Patron IDs

The script **WritePatronIdsFile.ksh** writes to a file the IDs (001 tag) of the patron records in the database. You can use the output file with any script or executable that accepts as its input a file of patron IDs.

7.1 Running WritePatronIdsFile.ksh

To extract a list of patron IDs,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type: **`WritePatronIdsFile.ksh [output filename]`**

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

   Note: If you specify the name of an existing file, the script will overwrite that file.

3. Press Enter.

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

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

8. Updating Temporary Circulation Counts

Each item record in your database has a temporary circulation count, which is the total number of times that the item has circulated since some specified date. Virtua provides a utility **ReSetTempCircCount.ksh** that lets you reset to zero the temporary circulation count of a set of specified items.

8.1 Running ReSetTempCircCount. sh

To update the temporary circulation count of item records,

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

2. At the prompt, type:

   ```
   ReSetTempCircCount.sh [input type] < [input file]
   ```

   Where . . .

   - **[input type]** specifies the type of input. You can type:

     - **1** - To specify that the input file contains a list of Item-IDs.
     - **2** - To specify that the input file contains a list of item barcodes.
     - **3** - To specify NO input file, in which case the temporary circulation count will be updated for ALL the items in your database.

   - **[input file]** specifies the name of the input file.

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

3. Press Enter.

   Virtua updates the temporary circulation count for the specified items. Additionally, the date from which the temporary circulation count is incremented is set to the current date.

9. Deleting Entries from the Circulation Transaction Log

Two scripts **Delete_Circ_Trans_Log_OFFLINE.sh** and **Delete_Circ_Trans_Log_ONLINE.sh** are available for use in deleting entries in a specified date range in the Circulation Transaction Log. Circulation entries are stored in the transaction log until they are deleted manually. You can run these scripts periodically to clean old transactions out of the transaction log.

The **Delete_Circ_Trans_Log_OFFLINE.sh** script is designed to delete a large number of circulation transaction log entries while the Virtua system is completely OFFLINE.

The **Delete_Circ_Trans_Log_ONLINE.sh** script is used to delete a small number of circulation transaction log entries, generally no more than one month of data, while the Virtua system is ONLINE.

9.1 Running Delete_Circ_Trans_Log_OFFLINE.sh

Important: Before running **Delete_Circ_Trans_Log_OFFLINE.sh**, shut down the **psdriver.exe**, Chamo, and **3mdriver.exe** and cease all other database activity. The system must be completely offline for this script to run. To increase the speed of running this script, ensure that the PGA memory allocated to the instance is set to 1G or higher. The PGA can be set to 1G by setting the memory_target to 2G and/or the pga_aggregate_target to 1G via the initvtls.ora. After adjusting the initvtls.ora parameters, you must restart the database.

To delete a large number of entries from the transaction log,

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

2. At the prompt, type:

```
Delete_Circ_Trans_Log_OFFLINE.sh startDate(YYYYMMDD)
endDate(YYYYMMDD)
```

2. Press Enter.

   The script calculates the number of transactions that will be deleted based on date range provided. The script displays the start date and end date you entered and the number of rows of transactions that it will delete from the transaction log.

   Note: The script deletes circulation transactions in an inclusive manner so that all transactions from the start date to the end date, including the start and end dates themselves, are deleted from the log. You can set the start date and end date to be the same day so that information from only that single date is deleted from the log.

   The script also displays a warning, reminding you that all database activity must be shut down and that there should be at least 30 percent of free space available in the small_tables tablespace so that the Circ_Transaction_Log table can be rebuilt.

   In addition, the script asks: "Have you run **$EXE_DIR/CheckTablespaceFileSize.sh** and verified that both UNDO tablespace and TEMP tablespace are at least 2G-4G in size?

3. In response, type:
   - **`Y`** to continue the script.
   - **`N`** to stop the script.

4. Press Enter. If you typed **`Y`** to continue, the script will prompt you with two more warnings/questions.

5. If you type **`Y`** again to indicate that you want to continue, the script will do the following:
   - Export the entire transaction log entries to a file named **exp_circ_transaction_data_13_09_2013_09_56_11.dat**

     Note: The current date and time are appended to the filename (before the extension).
   - Delete the information that falls within the date range you specified.

Note: The following InfoStation reports generate statistics from the transaction log:
- Basic Circulation Statistics
- Circulation By Call Number
- Journal Circulation

Entries that you delete from the transaction log will no longer appear in these reports.

9.2 Running Delete_Circ_Trans_Log_ONLINE.sh

Important: Running **Delete_Circ_Trans_Log_ONLINE.sh** consumes a lot of resources and takes more time than running **Delete_Circ_Trans_Log_OFFLINE.sh**, but its advantage is that it removes transactions while the system remains fully online and in use.

To delete a small number of entries from the transaction log,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type:
3. **`Delete_Circ_Trans_Log_ONLINE.sh startDate(YYYYMMDD) endDate(YYYYMMDD)`**
4. Press Enter.

   The script calculates the number of transactions that will be deleted based on date range provided. The script displays the start date and end date you entered and the number of rows of transactions that it will delete from the transaction log.

   The script also displays a warning, reminding you to be sure that system resources are sufficient and that the UNDO tablespace and REDO are adequately sized.

   Next, the script asks: "Do you want to start the deletion process which will export the table then delete the specified rows? (y/n)"

5. If you type **`Y`** to indicate that you want to continue, the script will do the following:
   - Export the entire transaction log to a file named **exp_circ_transaction_data_13_09_2013_09_56_11.dat**

     Note: The current date and time are appended to the filename (before the extension).
   - Delete the information that falls within the date range you specified.

10. Removing Identifying Patron Data from the Transaction Log

The script **update_circ_transaction_log.pl** lets you delete identifying patron information (patron IDs and patron barcodes) from the circulation transaction log entries. Circulation entries are stored in the transaction log until they are deleted manually. You can run **update_circ_transaction_log.pl** periodically to remove identifying patron information from the transaction log, or you can create a cron job to run the script at scheduled intervals.

Note: For check-out and request transactions, the **update_circ_transaction_log.pl** script only deletes identifying patron information for *completed* transactions.

- A check-out transaction is complete when the item is either checked in or deleted from the patron activity.
- A request transaction is complete when the request has been either satisfied or deleted.

The script **update_circ_transaction_log.pl** also populates the transaction log with several additional patron information data fields. Unless **update_circ_transaction_log.pl** is run, this data is not stored in the transaction log:

- Institution name
- College or school
- Department
- Major

Note: If the **update_circ_transaction_log.pl** script is NOT run, the above data is not stored in the transaction log since it can be retrieved using the stored patron ID.

10.1 Running update_circ_transaction_log.pl

To delete patron data from the transaction log,

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

2. At the prompt, type: **update_circ_transaction_log.pl**

3. The script will prompt you to include a start date and end date in YYYY-MM-DD HH:MI:SS format, which allows you to make an incremental update. This parameter is optional.

4. Press Enter.

   The **update_circ_transaction_log.pl** script does the following:

   - Removes identifying patron data from the circulation transaction log.
   - Sets the patron barcode values to null in the transaction log.
   - Sets the patron ID values to zero.
   - Populates the transaction log with the patron's institution name, college or school, department, and major.

Notes:
The following InfoStation reports generate statistics from the transaction log:

- Basic Circulation Statistics
- Circulation By Call Number
- Journal Circulation

Data that you delete from the transaction log will no longer appear in these reports.

If you wish to run **update_circ_transaction_log.pl** as a cron job, you can include the date range parameters as command-line arguments when running this script. To do this, you must use the following format:

```
update_circ_transaction_log.pl --start-date YYYY-MM-DD
HH:MI:SS --end-date YYYY-MM-DD HH:MI:SS
```

For example:

```
update_circ_transaction_log.pl --start-date 2011-01-01
--end-date 2013-01-01
```

While the start date and end date are optional, if you enter one you must enter the other.

11. Extracting Barcodes of Checked Out Items

The script **CheckedOutItems.sh** outputs the barcodes of all items in the database that are currently checked-out and not disputed or waived. The script outputs to standard output, which you can redirect to write to a file if you wish. You can use the file with any script or executable that accepts as its input a file of barcodes.

11.1 Running CheckedOutItems.sh

To extract a list of barcodes,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type: **`CheckedOutItems.sh > [file]`**

   Where **[file]** is the name of the file to which you want to redirect the standard output of barcodes.

   Note: If you specify the name of an existing file, the script will overwrite that file.

3. Press Enter.

   The script writes out the barcodes to the specified file.

12. Updating Due Dates

The interactive script **UpdateDueDate.sh**. updates the due dates of checked-out items in the database. You can run the script on a file of item IDs or on the entire database. The script offers prompts to specify a range of due dates to change, and the option to update the due dates only for items checked out at a specific location.

12.1 Running UpdateDueDate.sh

To update the due dates of a list of items,

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

2. At the prompt, type:

   ```
   UpdateDueDate.sh > [file]
   ```

   where **[file]** is the name of a line-delimited file of item IDs. The **[file]** value is optional.

3. Press Enter.

4. Answer the prompts regarding the starting and ending date and time for the due date range. If you want to change the due date of items due on one day, enter a start date and not an end date.

5. Answer the prompt regarding limiting the update to check-outs performed at a single location. If you enter **y** for yes, the script also prompts you to enter a location code.

6. Answer the prompt regarding the new due date. The script prompts with a message to confirm whether you want to update the database.

7. Enter **y** or **n** to confirm or cancel the database update. If you enter **y**, the script updates the due dates of the items that meet the entered criteria, and outputs a list of the updated items. Any items that had been considered overdue are no longer overdue, and any patron blocks associated with previously overdue items are removed.

13. Calculating Overdue Fines Prior to Check-in

By default, overdue fines are calculated and applied to the patron's account at the time that the overdue item is checked in. However, you can use the executable **UpdateOverdueFines.exe** to calculate fines for overdue items that have NOT yet been checked in.

13.1 Running UpdateOverdueFines.exe

To calculate overdue fines prior to check-in,

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

   The **UpdateOverdueFines.exe** executable assesses overdue fines for each overdue item in accordance with the fine rules for each item's shelving location in the Virtua Profiler. The new overdue fines are added to the patron's account, and are displayed on the Patron Information window > Account tab > Checked Out Overdue page. For more information on defining fine values in the Profiler, see the *Virtua Profiler/Circulation Parameters User's Guide*.

Note: Overdue fines are recalculated, if necessary, when the item is checked in in Virtua. When the item is checked in, the overdue fines are moved from the Checked Out Overdue page to the Overdue page on the Account tab. If the item is placed in Dispute and then checked in, the fine is moved from the Disputed page to the Overdue page on the Account tab.

14. Deleting Expired Patron Blocks

Library-defined blocks are stored in the 043 tag of the patron record. The subfield $d of the 043 contains the expiration date and time of the block. These blocks are entered either manually or automatically via the Convert Fine to Block functionality available from the Location + Patron + Item matrix in the Virtua Profiler.

By running the script **DeleteExpiredPatronBlocks.sh**, you can delete all expired blocks (i.e., 043 tags) from patron records.

Note: When the 043 subfield $d date/time has been reached, the server will no longer consider the patron blocked, so it is not necessary to run this script at short intervals to restore patron borrowing privileges.

14.1 Running DeleteExpiredPatronBlocks.sh

To delete all expired blocks from patron records in the database,

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

Virtua deletes all patron 043 tags that contain an expired date/time in the subfield $d, whether that date was entered manually or entered automatically by the system.

15. Enabling Floating Collections

Floating collections let you designate some items in your collection to "float"—that is, to be assigned the most recent check-in location as their shelving location (the items' owning location is maintained). This can help direct a library system's materials to the branches that use them the most. In addition, letting items float often means that items spend less time in transit and more time available for checkout.

The first step to enabling floating collections is to run the script **Schedule_FloatingMV_Refresh.sh**. When you run this script, you must determine the interval at which you want the floating information in your database (e.g., the Shelving Location listed on the Item Information window) to be updated. For most libraries, an interval of one hour is appropriate; refreshing this information more often may lead to a use of resources that could overload your server.

To run **Schedule_FloatingMV_Refresh.sh**,

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

2. At the prompt, type:

   `Schedule_FloatingMV_Refresh.sh [numeric_interval] [period_type]`

   Where **`[numeric interval]`** and **`[period type]`** define the interval of time that passes between refreshes.

For instance, to implement floating collections and update floating parameters within Virtua once per hour, type:

Schedule_FloatingMV_Refresh.sh 1 h

To update every 90 minutes, type:

`Schedule_FloatingMV_Refresh.sh 90 m`

This script accepts only "h" for hours and "m" for minutes as refresh period types. The interval is calculated based on the interval *between* refreshes. For instance, if the floating collection information begins an update at 5:00 that is completed at 5:10, and the refresh interval is one hour, the next floating collection update will begin at 6:10.

Note: Limits based on location and item class can be set on the floating items so that no branch ends up with too many or too few items on their shelves. See the *Virtua Profiler/Global Parameters User's Guide* and the *Virtua Profiler/Circulation Parameters User's Guide* for information about setting limits on floating collections. The final step to enabling items to float is to select the Floats check box on the Item Elements tab of the Item Information window in the Virtua client.

Important: The script **Schedule_FloatingMV_Refresh.sh** MUST be run for your library to use floating collections. If this script is not run, then Virtua will not be able to let items float.

16. Setting the Automatic Update of Shelving Location

When an item with an At Shelving Location Until date is checked in after this date, Virtua's default behavior is to automatically reset the shelving location to the owning location and notify the staff user.

If you do not want Virtua to automatically reset the shelving location, you can run the script **SetAutoShelfLocationReset.sh** to change this behavior so that the shelving location is not changed when an item is checked in with an At Shelving Location Until date that has passed.

Usage:

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

2. To *disable* Virtua's default behavior of resetting the shelving location of an item at check-in when the At Shelving Location Until date has passed, at the prompt, type:

   ```
   SetAutoShelfLocationReset.sh 0
   ```

   -OR-

   To *re-enable* Virtua's behavior of resetting the shelving location of an item at check-in when the At Shelving Location Until date has passed, type:

   ```
   SetAutoShelfLocationReset.sh 1
   ```

3. Press Enter.

Virtua changes its shelving behavior based on your setting.

Note: For more information on the At Shelving Location Until date, see the *Cataloging Workflow-based Reference Guide*.

17. Removing the Street Date Status

When you want to catalog an item before it can be released to the public, i.e., before its street date, you can store the street date for the item in YYYYMMDD format in the 369 tag subfield $a of the bibliographic record. This enables you to use the Profiler's Circulation Basic Options setting *Not available until street date status* to assign a certain status to items added to a bibliographic record with a street date that is in the future. Once that date is passed, you must run the script **RemoveStreetDateStatus.sh** to remove the pre-street date status from any items attached to bibliographic records with street dates in the past. The items can then be available for circulation.

Since books, movies, and music are generally released at least once per week, if you use the *Not available until street date status* setting, you will want to set up a cron job to run **RemoveStreetDateStatus.sh** on a regular basis. For information on running a cron job, see the *System Management Reference Guide*.

To remove statuses from items attached to bibliographic records with street dates that have passed,

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

2. At the prompt, type: **`RemoveStreetDateStatus.sh`**

3. Press Enter.

If the *Not available until street date status* check box is selected in the Circulation Basic Options, Virtua will delete the statuses of items attached to bibliographic records with a date in the 369 tag subfield $a that is in the past. For information on the *Not available until street date status* parameter, see the *Virtua Profiler/Circulation Parameters User's Guide*.

18. Updating the Patron Type by Age

The script **UpdatePatronTypeByAge.sh** will change a patron's type if the patron meets the age threshold specified in the Patron Types parameter in the Virtua Profiler.

On the Patron Type window in the Profiler, if you enable the Update Patron Type option, specify the age at which the change will take place, and select the succeeding patron type, then **UpdatePatronTypeByAge.sh** will find those patrons that are set to be automatically updated and perform the update. The script will log the number of updates it made.

You will want to set up a cron job to run this script on a regular basis. For information on running a cron job, see the *System Management Reference Guide*.

To update patron types by age,

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

2. At the prompt, type: **`UpdatePatronTypeByAge.sh`**

3. Press Enter.

For information on the Update Patron Type option, see the *Virtua Profiler/Circulation Parameters User's Guide*.

19. Updating the Patron Type

The script **RestorePatronsToGoodStanding.sh** identifies eligible patrons or changes patron records from one patron type to another based on a set of criteria. This script may be useful for updating the patron type of recently registered (probationary) patrons who have been assigned a patron type with limited privileges. (See also the InfoStation report Update Probationary Patrons, which invokes the **RestorePatronToGoodStanding.sh** script.)

The set of criteria used by the script consists of: 1) A specified number of days since the date of registration and 2) Probationary patrons with no overdue items, billed items, or blocks.

Note: The script will change the patron type of only those patrons who have no overdue items, billed items, or blocks.

19.1 Running the Script

To run RestorePatronsToGoodStanding.sh,

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

2. At the prompt, type:

   ```
   RestorePatronsToGoodStanding.sh [xx] [update?]
   [current patron type]
   [new patron type]
   [current patron type]
   [new patron type]
   ...
   ```

   Where **[xx]** is the number of days since registration, **[current patron type]** is the patron type to be changed, and **[new patron type]** is the patron type to change to. You can enter up to 10 pairs of old and new patron types.

   -AND-

   Where [**update?**] defaults to **Y**(es) if nothing is entered on the command line.

   - If you type a "Y" or a "1," the database will be updated.
     (The script will update the patron type of patrons in the database who fit the criteria.)

- If you type any value other than a "Y" or a "1," the database will NOT be updated.

3. Press Enter.

The script will identify the patron type of patrons in the database who fit the criteria, and, if appropriate, update the patron records.

Command-line Example:

```
RestorePatronsToGoodStanding.sh 30 N
AP
AD
JP
JU
```

19.2 Logging Results

Regardless of whether the database is updated, the **RestorePatronsToGoodStanding.sh** script logs data in a comma-delimited file **RestorePatronsToGoodStanding [timestamp].log** as follows:

- **If the database is updated, the script logs the following information about each patron that qualifies:**
  Updated, current date, patron ID, start (registration) date, old patron type, new patron type

- **If database is NOT updated, the script logs the following information about each patron that qualifies:**
  ToBeUpdated, current date, patron ID, start (registration) date, old patron type, new patron type

- **If a patron does NOT qualify to have their patron type changed, the script logs the following information about the patron:**
  Not Updated, current date, patron ID, start (registration) date, old patron type

20. Assigning Request Groups to Requests

The script **PopulateRequestGroup.sh** attempts to assign a request group to each request in the database that does not have one assigned and log information about those requests for which no group could be chosen. The script ignores requests that are already assigned a group.

To assign request groups to requests,

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

2. At the prompt, type: **`PopulateRequestGroup.sh <output_file_name>`**

   Where **<output_file_name>** is the name of the file where information about requests for which a request group could not be chosen will be logged.

3. Press Enter.

   - If the pickup location has a request group defined, the script will assign that request group to the request.
   - If the pickup location does not have a request group defined, the script will assign the first request group that the location (where the request was placed) is a member of.

Note: It's safe to run this script multiple times. For instance, if the script fails to update a large number of requests, you could use the Virtua Profiler to assign a request group to the locations where those requests were placed. Then you could execute the script again.

For information on request groups, see the *Virtua Profiler/Circulation Parameters User's Guide*.

21. Removing Invalidated Requests

When the InfoStation Expired Requests and Unsatisfied Request Cancellation reports run, they invalidate expired requests, rather than delete them. In most cases, this is desirable, as there may be times when the library may want to reinstate these requests. However, items cannot be deleted from Virtua when the item is still associated with these invalidated requests.

The script **RemoveInvalidatedRequests.sh** deletes invalidated requests from the database, which means that items associated with these invalidated requests can then be deleted.

To remove invalidated requests,

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

2. At the prompt, type: **`RemoveInvalidatedRequests.sh <number of days>`**

   Where **<number of days>** is the number of days that have passed since the request was invalidated.

3. Press Enter.

Virtua deletes all requests that were invalidated at least **<number of days>** before the day the script is run.

For information on the Expired Requests and Unsatisfied Request Cancellation reports, see the *InfoStation/Circulation Reports Reference Guide*.

22. Reporting on All Requested Items

With regard to the InfoStation Recalled Overdues report, Recalled Bills report, and Alerts and Blocks for recalled overdue items, Virtua's default behavior is to report on/consider only *recalled* overdue items; Virtua ignores *requested* overdue items.

If you want Virtua to report on/consider ALL requested overdue items, not just those that have been recalled, you can run the script **SetConsiderAllRequestedItemsFlag.sh** to change the default behavior.

When run, the script can set system parameter 180 on (1) so that the Recalled Overdues report, Recalled Bills report, and the portion of the Alerts and Blocks Matrix that deals with Recalled Overdue items will treat ALL requested overdue items as recalled overdue items.

Usage:

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

2. Do one of the following:
   - To *disable* Virtua's default behavior of reporting on/considering only recalled overdue items in the functions described above, at the prompt, type:
     `SetConsiderAllRequestedItemsFlag.sh 1`

   - To *re-enable* Virtua's behavior of reporting on/considering only recalled overdue items in the functions described above, at the prompt, type:
     `SetConsiderAllRequestedItemsFlag.sh 0`

3. Press Enter.

   Virtua changes its behavior based on your setting.

Note: For more information on the Recalled Overdues report and Recalled Bills report, see the *InfoStation/Circulation Reports Reference Guide*. For more information on the Recalled Overdue aspect of the Alerts and Blocks Matrix, see the *Virtua Profiler/Circulation Parameters User's Guide.*

23. Reprocessing Requests to Reset Request Types

After a migration or any other event that has changed request types in the database, your institution may want to refresh, or reprocess, those requests that have had their types changed. The executable **RequestRefresh.exe** allows you to reset request types in batch. When you run the executable, you can choose to update either all pending requests in the database or a list of specific request IDs provided on the command line or via a file. The program updates the request types per the Auto configuration in the Virtua Profiler. Optionally, you can specify a new request type.

To reset the request types in the database,

1. Log in to your server as **dbadmin**.
2. Change directory to EXE_DIR.
3. At the prompt, type: **`RequestRefresh.exe [options]`**

   Where **[options]** can be replaced by one of the following:

   -h [--help] - Print usage statement.

   -r [--input-requests] arg (=1) - Specify which requests get processed:
   1 - Input Request IDs
   2 - Process All Unsatisfied Requests

   -t [--new-request-type] arg - Specify new request type:
   0 - Auto
   1 - Hold
   2 - Recall
   3 - Loan
   4 - Page

   -i [--institution-symbol] arg - Specify institution symbol. Only requests placed at this institution will be processed.

   Note: Option **-i** appears only for customers in a consortium environment. This option is used to specify the current institution. If you want to update all pending requests, only requests placed at the institution represented by the symbol will be processed.

RequestRefresh.exe will create a new log file each time it is run. The file is written to the directory from which you invoke the executable ($PWD). The name of the file contains the current date/time and is in the following format:

```
RequestRefresh_YYYYMMDD_HH24MISS.log
```

The log file lists the requests that were processed and whether or not the "refresh" was successful. To get more information about why specific requests failed to be refreshed, turn on V_LOG_LEVEL.

Example 1:

```
$EXE_DIR/RequestRefresh.exe -r 2 -t 0
```

This command is the equivalent of opening every single pending request in the client and modifying it to have a type of Auto. To be included for processing when the **-r 2** option is specified, the following must be true for a particular request:

- Have a status of Pending
- Not be expired
- Not have an item held
- Not be invalidated

Example 2:

```
$EXE_DIR/RequestRefresh.exe -r 1 < file_of_request_ids
```

Where **file_of_request_ids** is a file containing a list of requests that need to be refreshed. This command will refresh each of the specified requests without making any changes to their type. As an example of refreshing, Virtua may need to choose a new (single) item to receive the Requested for Loan status for a request.

24. Changing the Requested for Loan Status on Requested Items

When you deactivate a request by clicking the Deactivate button on the Pending page of the Patron Information window's Activity tab, Virtua deactivates the request and prompts you to specify a period of time during which the request is considered inactive. If an item has been assigned to satisfy the request but has not yet been trapped, the item retains the Requested for Loan status, even though the request is no longer active.

Running the **HandleRequestDeactivationStatus.sh** script identifies requests that have been deactivated and removes the Requested for Loan status from items that have been assigned (but not yet trapped) to satisfy the deactivated requests. Items from which the Requested for Loan status is removed are then considered available to satisfy other requests.

Running the **HandleRequestDeactivationStatus.sh** script also identifies requests that have been reactivated, either manually or because the deactivation period has expired, and adds the Requested for Loan status to items that had previously been assigned to satisfy the reactivated requests, as long as these items qualify to satisfy the request.

To remove or restore the Requested for Loan status from untrapped items,

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

2. At the prompt, type: **`HandleRequestDeactivationStatus.sh`**

3. Press Enter.

Virtua identifies deactivated and reactivated requests and changes the Requested for Loan status on items to the appropriate status.

For information on deactivating and reactivating requests, see the *Circulation Workflow-based Reference Guide*.

25. Appendix A - Scripts and Executables Discussed in this Guide

In the following table, we list and briefly describe each script and executable discussed in this user's guide. For additional information on each program, in the first column, click the filename, which is hyperlinked to the appropriate section in this guide.

Filename	Purpose	Command line Parameters
CheckCircParameters.sh	Examines your circulation parameters and reports any discrepancies. This script reports any problems with your circulation parameters in the file **CheckCircParameters.log**.	None
CheckedOutItems.sh	Outputs the barcodes of all items in the database that have been checked out.	The name of a file to which the script will write item barcodes
Delete_Circ_Trans_Log_OFFLINE.sh	Deletes a large amount of circulation transaction log data while the Virtua system is completely *offline*. **Note:** Transactions that you delete will no longer appear in InfoStation reports that query the transaction log.	Start and end dates.
Delete_Circ_Trans_Log_ONLINE.sh	Deletes a small number of circulation transaction log entries, generally no more than one month of data, while the Virtua system is *online* and in use. **Note:** Transactions that you delete will no longer appear in InfoStation reports that query the transaction log.	Start and end dates.

Filename	Purpose	Command line Parameters
DeleteExpiredPatronBlocks.sh	Deletes expired blocks (043 tags) from patron records in the database.	None
DeleteExpiredPatrons.sh	Generates a list of patron IDs and deletes the patron records (unless otherwise specified) based on the expiration date of the patron records.	The expiration date of the patron record and, for consortium databases only, the institution symbol.
DeleteInactivePatrons.sh	Generates a list of patron IDs and deletes the patron records (unless otherwise specified) based on one of the following two criteria: • Last activity date of the patron record. • Length of time (in months or years) that the patron record has been inactive	The script is interactive. Consortium databases only must specify the institution symbol.
DeleteOldFines.sh	Deletes fines and fees assessed before a given date, and outputs a file of information about the deleted fines and fees.	None. This is an interactive script.
DeletePatronPasswords.sh	Deletes unencrypted patron passwords in subfield $b of the 015 and 016 tags from all patron records that have the new encrypted version in subfield $c	None
DeletePatrons.sh	Deletes patron records, regardless of their outstanding associations, based on a file of patron IDs. **Important:** Since this script deletes patron records, it is very important that you have a recent export of the database or of your patron records.	The name of a file of patron IDs and, for consortium databases only, the institution symbol.

Filename	Purpose	Command line Parameters
ExportPatronBarcodeTable.sh (*Not discussed in this guide*)	Exports the patron_barcode table so that **Update_Patron_015b.sql** can be run successfully.	Usage: `ExportPatronBarcodeTable.sh <DataDirectory>` where <DataDirectory> = name of the directory where the export files will be written.
HandleRequestDeactivationStatus.sh	Identifies requests that have been deactivated and removes the Requested for Loan status from items that have been assigned (but not yet trapped) to satisfy the requests, AND identifies requests that have been reactivated, and adds the Requested for Loan status to items that had been previously assigned to satisfy the request.	None
PopulateRequestGroup.sh	Assigns a request group to each request in the database.	None
RemoveInvalidatedRequests.sh	Removes invalidated requests from the database.	Number of days that have passed since the request was invalidated
RemovePatronsByDeletionDate.sh	Generates a list of patron IDs and deletes the patron records (unless otherwise specified) whose deletion date has passed	Consortium databases only must specify the institution symbol.
RemoveStreetDate.sh	Removes the statuses of items attached to a bibliographic record with a date that has passed in the 369 tag subfield $a.	None
RequestRefresh.exe	Reprocesses or refreshes requests whose types have been changed due to a migration or other event.	Optional options are available.
ReSetTempCircCount.sh	Resets to zero the temporary circulation count of item records.	The input type and the name of the file of Item-IDs or barcodes.

Filename	Purpose	Command line Parameters
RestorePatronsToGoodStanding.sh	Changes patrons of one patron type to another patron type.	Number of days since patron registration, old patron type, new patron type.
Schedule_FloatingMV_Refresh.sh	Enables floating collections and defines the interval at which floating information will be updated.	Update interval
SetAutoShelfLocationReset.sh	Disables or enables the reset of shelving location when checking in items whose At Shelving Location Until date has passed.	**0** - Toggle off the automatic update of shelving location. **1** - Toggle on the automatic update of shelving location.
SetConsiderAllRequestedItemsFlag.sh	Disables or enables Virtua's consideration of all requested overdue items as recalled overdue items for the sake of the Recalled Overdues report, Recalled Bills report, and the Alerts and Blocks Matrix.	**0** - Toggle off Virtua's consideration of all requested overdue items as recalled overdue items. **1** - Toggle on Virtua's consideration of all requested overdue items as recalled overdue items.
UpdateDueDate.sh	Updates due dates for items checked out within a date range and optionally at a specified location.	None. This is an interactive script.
UpdateOverdueFines.exe	Calculates overdue fines for overdue items that have NOT yet been checked in.	None
UpdatePatronTypeByAge.sh	Change a patron's type if the patron meets the age threshold specified in the Patron Types parameter in the Virtua Profiler.	None
update_circ_transaction_log.pl	Deletes identifying patron information (patron IDs and patron barcodes) from the circulation transaction log entries.	None
Update_Patron_015b.sh	Adds or updates tag 015, subfield $b for the patrons specified in a file of Patron-IDs.	None. The script prompts the user for input.

Filename	Purpose	Command line Parameters
WritePatronIdsFile.ksh	Writes to a file the Patron-ID (001 tag) of every patron record in the database.	The name of a file to which the script will write record IDs.

26. Appendix B - Circulation Transaction Type Codes

Virtua uses the following transaction type codes to store information in the CIRC_TRANSACTION_LOG database table. These codes may be of use to you if you are authorized to use SQL queries to retrieve transaction log data from the database.

- 1 - Regular check-out
- 2 - In-house check-out
- 3 - Fixed Time and Date check-out
- 4 - Selected Time and Date check-out
- 5 - not used
- 6 - Regular renewal
- 7 - In-house renewal
- 8 - Fixed Tim and Date renewal
- 9 - Selected Time and Date renewal
- 10 - not used
- 11 - Regular check-in
- 12 - Selected Time and Date check-in
- 13 - Added request
- 14 - Modified request
- 15 - Deleted request
- 16 - Resubmitted request
- 17 - Satisfied request
- 18 - Denied request
- 19 - Added item booking request
- 20 - Modified item booking request
- 21 - Deleted item booking request
- 22 - Added document delivery request
- 23 - Modified document delivery request
- 24 - Deleted document delivery request
- 25 - Disputed fine or fee
- 26 - Waived fine or fee
- 27 - Reinstated disputed item
- 28 - Review notice issued
- 29 - Overdue notice (non-recalled item) issued

- 30 - Recalled overdue notice issued
- 31 - Pre-billing search list
- 32 - Pre-billing search list (requested)
- 33 - Bill issued
- 34 - Bill issued (requested)
- 35 - Recall letter issued
- 36 - Request letter issued
- 37 - Recall cancel letter issued
- 38 - Check-in denied at location
- 39 - Manual bill issued

27. Appendix C - Changes in this Guide

27.1 Changes for Version 16.1

No changes were made.

Index

F

H

I

J

L

M

N

O

P

R

S

T

U

V

W