We find this closed branch process fairly straightforward for managing various aspects of a temporary or even less temporary closure.

The instructions below are for utilizing the closed branch process in our system, and reference customized jobs that aren’t provided. However, it should provide enough context to establish in your own system.

**Note:** we extend checkouts and hold until dates by 2 weeks, but this can be customized in the ‘Maintenance’ file.

SILS Steps for Branch Closures

***Note:****All tables and stored procedures below are in the Polaris database, and use the Polaris schema (i.e. have “Polaris.” before their names).*

* **Main Process Table:**  
  SILS\_ClosedBranch\_List
  + Columns:
    - **OrgID** - OrganizationID of the closed branch/library
    - **OrgCodeID** - the level of the closed org (branch = 3, library = 2)
    - **Exception** - Used to ID open branches under a mostly closed library
    - **HoldInactivation** - Holds should be inactivated for the closed org
    - **HoldUntilDates** - Hold Unclaimed dates should be extended
    - **DueDates** - Checked-out item due dates should be extended
    - **DatesClosed** - Branch hours should be closed, stopping picklists
    - **Restored** - The branch will be opened when the next restore job runs
  + To view: right-click > select top 100 rows
  + To edit: right-click > edit top 200 rows
  + With the exception of the OrgID and OrgCodeID, all the fields on the table are of a ‘bit’ type that uses 1 & 0 values to equal True and False (respectively)
* **Additional Process Components:**
  + **Table:**  SILS\_ClosedBranch\_OriginalHours
    - Stores original hours of operation for a branch if ‘DatesClosed’ in the main table is set to True
  + **SProc:**  SILS\_ClosedBranch\_Maintenance
    - Job:  SILS Daily Table Updates
    - Runs daily at 6 AM (SK)
    - 3 tasks (for branches with pertinent fields in the main table set to True):
      * Sets branch hours of operations to closed, and backs up original open hours
      * Extends checkout due dates
      * Extends hold unclaimed dates
  + **SProc:**  SILS\_ClosedBranch\_Restore
    - Job:  SILS - Closed Branch Restore
    - Runs daily at 3 PM (SK)
    - 3 tasks (for branches with ‘Restored’ in the main table set to True):
      * Restores original branch hours, if backed-up
      * Sets inactivated holds to reactivate the next day, if holds inactivated for the branch
      * Removes the branch(es) from the main process table; if a closed library is set to restore (incl. any exception branches), the process will also remove the library and exception branches
  + **SProc:**  SILS\_Holds\_Inactivations
    - Job:  SILS Delay Holds Setting
    - Every 10 minutes, between 7:15 AM and 9:15 PM (SK)
    - Inactivates active holds for branches with ‘Hold Inactivation’ set to True in the main table

Branch Closing

1. Which branch(es) are closing
   1. Create row in Closed Branch table for branch **OrgID**, **OrgCodeID**
2. Which (if any) notices should be turned off
   1. Turn off/on notice types
   2. Branch > Parameters > Notifications
3. Should hold picklists be stopped:  *yes/no*
   1. If yes, in Closed Branch table, set **DatesClosed** to ‘1’ (True)
   2. When the morning process runs, the existing hours of operation for the branch (if not all showing as closed already), will be backed-up on the ‘Original Hours’ table, and all days will be set to closed for the branch.
4. Should active holds be inactivated:  *yes/no*
   1. If yes, in Closed Branch table, set **HoldInactivation** to ‘1’ (True)
5. Should checkout due dates be extended:  *yes/no*
   1. If yes, in Closed Branch table, set **DueDates** to ‘1’ (True)
6. Should held hold unclaimed dates be extended:  *yes/no*
   1. If yes, in Closed Branch table, set **HoldUntilDates** to ‘1’ (True)
7. The ‘Exception’ field
   1. This field should always have a value of 0 (False), except in cases where a library is closing with some branches remaining open.  See the ‘Library Closing’ section below.

Library Closing

If a library is fully or mostly closing, it can also be managed through the ‘closed branch’ process:

1. Create row in Closed Branch table for the library OrgID, with a OrgCodeID value of ‘2’
2. If there are any branches in the library remaining open, they can be included as an **Exception**.  Add these branches in their own rows on the main process table, including an OrgCOdeID of ‘3’, but in the ‘Exception’ field set the value to ‘1’ (True), and leave the other table fields blank.
3. For the library row in the process table, add values to the remaining columns per the branch closing settings above.

Hold placed/denied language in PAC:

In WebAdmin, load ILL strings. Edit ILL\_TEXT\_REQNOTPERMITTED

Branch/Library Re-opening

1. Schedule the date the branch is reopening
   1. Create calendar reminders
   2. Day before re-opening, in Closed Branch table, add ‘1’ (True) in the Restored column
      1. Restore script runs at 3pm daily
2. Confirm the date the request picklist should be restarted, if different from the reopening date
   1. If prior to full re-opening, you can set the branch to restore on an earlier date, and then re-add the branch to the main process table with any continuing closed settings added back.
3. Verify which notices should be turned on - in most cases these will mirror your library’s other branches
   1. Turn notification types back on
4. Establish whether notices should be sent for currently held holds
   1. If yes, execute script to repopulate holds notices (script below)
   2. If queued notice counts match originals, uncomment out, highlight, and execute Commit

Hold Notice Repopulation Script

USE POLARIS

DROP TABLE IF EXISTS #Libs

DROP TABLE IF EXISTS #LibList

CREATE TABLE #Libs (OrganizationID int, OrganizationCodeID int)

INSERT INTO #Libs (OrganizationID, OrganizationCodeID)

VALUES (82,3) -- replace with relevant branch ID

CREATE TABLE #LibList (OrganizationID int, OrganizationCodeID int)

INSERT INTO #LibList

SELECT o.OrganizationID, o.OrganizationCodeID

FROM Polaris.Organizations o (NOLOCK)

LEFT JOIN #Libs l

    on o.OrganizationID = l.OrganizationID and l.OrganizationCodeID = 3

WHERE o.OrganizationCodeID = 3

AND (l.OrganizationID is not null

OR o.ParentOrganizationID IN

    (SELECT

        l.OrganizationID

    FROM #Libs l

    WHERE l.OrganizationCodeID = 2

    ))

--  Hold notice counts overall

SELECT

COUNT(distinct shr.SysHoldRequestID) "All\_Notices\_Sending",

COUNT(distinct shr2.SysHoldRequestID) "Notices\_Re-Sending"

FROM Polaris.SysHoldRequests shr

INNER JOIN #LibList ll

    on shr.PickupBranchID = ll.OrganizationID

LEFT JOIN Polaris.SysHoldRequests shr2

on shr.SysHoldRequestID = shr2.SysHoldRequestID and shr2.HoldNotificationDate is not null

LEFT JOIN Polaris.PatronRegistration pr (nolock)

on shr.PatronID = pr.PatronID and pr.ExcludeFromHolds = 1

WHERE shr.SysHoldStatusID = 6

and pr.PatronID is null

and shr.TrappingItemRecordID is not null

--  Counts by library

SELECT

o.name "Library",

COUNT(distinct shr.SysHoldRequestID) "All\_Notices\_Sending",

COUNT(distinct shr2.SysHoldRequestID) "Notices\_Re-Sending"

FROM Polaris.SysHoldRequests shr

INNER JOIN #LibList ll

    on shr.PickupBranchID = ll.OrganizationID

INNER JOIN Polaris.Organizations o (nolock)

on shr.PickupBranchID = o.OrganizationID

INNER JOIN Polaris.Organizations op (nolock)

on o.parentOrganizationID = op.OrganizationID

LEFT JOIN Polaris.SysHoldRequests shr2

on shr.SysHoldRequestID = shr2.SysHoldRequestID and shr2.HoldNotificationDate is not null

LEFT JOIN Polaris.PatronRegistration pr (nolock)

on shr.PatronID = pr.PatronID and pr.ExcludeFromHolds = 1

WHERE shr.SysHoldStatusID = 6

and pr.PatronID is null

and shr.TrappingItemRecordID is not null

GROUP BY o.name

BEGIN TRANSACTION

UPDATE shr

SET HoldNotificationDate = NULL, HoldNotification2ndDate = NULL

FROM Polaris.SysHoldRequests shr

INNER JOIN #LibList ll

    on shr.PickupBranchID = ll.OrganizationID

WHERE SysHoldStatusID = 6

and shr.TrappingItemRecordID is not null

--Re-populate hold notices to the queue

DECLARE @nItemRecordID INT,

        @nNotificationTypeID INT,

        @nPatronID INT,

        @nBranchID INT

DECLARE HoldNotices CURSOR FOR

SELECT

    shr.TrappingItemRecordID,

    2,

    shr.PatronID,

    shr.pickupBranchID

FROM Polaris.Polaris.SysHoldRequests shr (NOLOCK)

INNER JOIN #LibList ll

    on shr.PickupBranchID = ll.OrganizationID

WHERE shr.SysHoldStatusID = 6

AND shr.HoldNotificationDate IS NULL

and shr.TrappingItemRecordID is not null

OPEN HoldNotices

FETCH NEXT FROM HoldNotices INTO @nItemRecordID, @nNotificationTypeID, @nPatronID, @nBranchID

WHILE @@FETCH\_STATUS = 0

BEGIN

    EXEC Polaris.AddNotification @nItemRecordID, @nNotificationTypeID, @nPatronID, @nBranchID

    FETCH NEXT FROM HoldNotices INTO @nItemRecordID, @nNotificationTypeID, @nPatronID, @nBranchID

END

CLOSE HoldNotices

DEALLOCATE HoldNotices

-- COMMIT TRANSACTION

-- ROLLBACK TRANSACTION

--  Validation - check queued notices match original counts

SELECT

o.name "Library",

COUNT(distinct nq.NotificationQueueID) "Notices\_Queued"

FROM Polaris.SysHoldRequests shr

INNER JOIN #LibList ll

    on shr.PickupBranchID = ll.OrganizationID

INNER JOIN Polaris.Organizations o (nolock)

on shr.PickupBranchID = o.OrganizationID

INNER JOIN Polaris.Organizations op (nolock)

on o.parentOrganizationID = op.OrganizationID

LEFT JOIN Polaris.SysHoldRequests shr2

on shr.SysHoldRequestID = shr2.SysHoldRequestID and shr2.HoldNotificationDate is not null

LEFT JOIN Polaris.PatronRegistration pr (nolock)

on shr.PatronID = pr.PatronID and pr.ExcludeFromHolds = 1

LEFT JOIN results.polaris.NotificationQueue nq (nolock)

on shr.PatronID = nq.PatronID and shr.TrappingItemRecordID = nq.ItemRecordID and nq.IsAdditionalTxt != 1

WHERE shr.SysHoldStatusID = 6

and pr.PatronID is null

and shr.TrappingItemRecordID is not null

GROUP BY o.name

DROP TABLE IF EXISTS #Libs

DROP TABLE IF EXISTS #LibList