User Centric Improvements with ILLiad and NCIP

Bill Jones
Creative Technologist

Leila Smith
Manager of Resource Sharing Operations & Training

IDS Project Conference 2014 | July 31, 2014 | Syracuse, NY
Background

• Harvard Library reorganization 2012
• Centralization of Resource Sharing services
• Centralized workflows = looking for efficiencies
Current State

- Borrowing and Lending Operations
  - BHA (Harvard-Andover Theological Library)
  - FLL (Loeb Design Library)
  - HBS (Harvard Business School)
  - HLS (Harvard College)
  - HMG (Graduate School of Education)
  - HMS (Countway Medical Library)
  - HMY (Harvard-Yenching Library) lending only
  - HMZ (Ernst Mayr Library)
  - HVL (Harvard Law Library)
  - KSG (Kennedy School Library)
  - DDO (*Dumbarton Oaks Research Institution*)
    - Washington, D.C.
  - HTV (Villa I Tatti)
    - Italy
- Lending only
  - CLS (Cabot Science Library)
  - HUL (Harvard Depository)
  - MCS (Sciences – Physics/Chemistry)
Future State?

- Borrowing/Lending
  - HUL (Harvard University)
  - HVL (Law)
  - HMS (Medical)
  - DDO (Dumbarton Oaks)
Patron Benefits

• Patrons select pick up location other than home library
• Return to any Harvard Library
  – Overdue notices
• One place to view and renew checked out items
• Standard due date/renewal policy for circulating items
• More items circulate
  – Foreign
  – Multi-volume
Staff Benefits

- “I never checked out this item!”
- Less confusion about renewal process
- Track returns to other Harvard Libraries
- Lost item charges
- Fewer steps in lending workflow
Planning

• Bookstraps vs slips and their design
  – Slips save time (and paper)
  – Bookstraps are not as easily lost, align with Borrow Direct service

• Email and print templates in Aleph
  – Can’t distinguish between in-library-use only and circulating
  – Minimal wording

• Print templates in ILLiad
  – Word template mail merge
  – If-then statements (TN-) (LU-)
Unlocking innovations to solve resource sharing’s biggest problems.

Bill Jones
Creative Technologist
Types of Addons

Client Side Addon
• Installed in the ILLiad\Addons directory
• Run within the Client using a ribbon and buttons
• Error logs appear in C:\Users\username\Documents\ILLiad\Logs\ILLiadClient.txt
• Refreshed using Manage Addons -> Reset Cache

System Addon
• Installed in the ILLiad\Addons directory
• Run within the Client through Registered Event Handlers
• Error logs appear in C:\Users\username\Documents\ILLiad\Logs\ILLiadClient.txt
• Refreshed by restarting ILLiad

Server Addon
• Installed through Customization manager and saved in a database
• Run outside of the Client on the server at specific time intervals
• Error logs appear in C:\Users\username\Documents\ILLiad\Logs\CustomizationManager.txt
• Refreshed by increasing the version number in the Config window
Borrowing Functions: Syncing Two Systems

Receiving Borrowed item at Harvard
- AcceptItem Call
- “Check In” button is clicked from “Check In From Lending Library”
- Creates minimal record in Aleph from ILLiad transaction information
- Creates Aleph Item Record and Live Hold for Patron

Item Returned by Harvard Patron
- CheckInItem Call
- “Process Queue” is clicked during “Check In Item”
- Allows for Auto Processing for multiple transactions
- Discharges item from patron in Aleph
Lending Functions: Syncing Two Systems

After Item is Pulled From Stacks
- CheckOutItem Call
- "Mark Found" button is clicked during "Update Stacks Searching"
- Checks item out to pseudopatron in Aleph

Item is Returned by Borrowing Library
- CheckInItem Call
- "Process Queue" is clicked during "Lending Returns"
- Allows for Auto Processing for multiple transactions
- Discharges item from pseudopatron (borrowing library) in Aleph
function Init()
RegisterSystemEventHandler("Name_of_event_handler", "name_of_function");
end

("Name_of_event_handler", "name_of_function");

These are very specific ILLiad names tied to very specific button clicks

These are not NCIP specific

These are not specific. You can customize these names.
So Which Registered Event Handlers?

**BorrowingRequestCheckedInFromLibrary**
This will trigger whenever a non-cancelled transaction is processed from the Check In From Lending Library batch processing form using the Check In, Check In Scan Now, or Check In Scan Later buttons.

**BorrowingRequestCheckedInFromCustomer**
This will trigger whenever an item is processed from the Check Item In batch processing form, regardless of its status (such as if it were cancelled or never picked up by the customer).
So Which Registered Event Handlers?

LendingRequestCheckOut
This will trigger whenever a transaction is processed from the Lending Update Stacks Searching form using the Mark Found or Mark Found Scan Now buttons. This will also work on the Lending Processing ribbon of the Request form for the Mark Found and Mark Found Scan Now buttons.

LendingRequestCheckIn
This will trigger whenever a transaction is processed from the Lending Returns batch processing form.
Understanding the XML Messages from ILLiad

**BorrowingRequestCheckedInFromLibrary**
- Check In From Lending Library
- Electronic Delivery
- Print
- Receive Requests
- Contact Customers

**AcceptItem Scheme**

**BorrowingRequestCheckedInFromCustomer**
- Check Out
- Check In
- Overdues
- Circulation

**CheckedIn Scheme**

**LendingRequestCheckIn**
- Remove Item
- Clear Processed Options
- Process Queue
- Auto Process

Slightly different based on the Item Identifier

**LendingRequestCheckOut**
- Cancel Request
- Conditionalize Request
- Mark Found
- Mark Found Scan Now

**CheckedOut Scheme**
XML: AcceptItem Call

49 lines for the message

Easy Breakdown:
* From Unique Agency
* To Unique Agency
Pickup Location (pulled from pre-defined .txt list)
* Unique Request ID/Unique Agency ID
Request Identifier Value (configurable TN#)
* Requested Action Type
* Unique Agency ID
Unique User (User ID, Pulled from Transaction)
Unique Item (configurable TN#)
Bibliographic Description (Author & Title, Pulled from Transaction)

* Set in Addon Config
AcceptItem: Setting the Pick up Location

local pickup_location_full = GetFieldValue("Transaction", "Location");
local sublibraries = assert(io.open("c:\Program Files\ILLiad\Addons\IDS_NCIP_Client\sublibraries.txt", "r"));
if sublibraries == nil then
    sublibraries = assert(io.open("c:\Program Files (x86)\ILLiad\Addons\IDS_NCIP_Client\sublibraries.txt", "r"));
end
local pickup_location = "";
local templine = nil;
if sublibraries ~= nil then
    for line in sublibraries:lines() do
        if string.find(line, pickup_location_full) ~= nil then
            pickup_location = string.sub(line, line:len() - 2);
            break;
        else
            pickup_location = "nothing";
        end
    end
    sublibraries:close();
end
Configurable Transaction Prefixes

Transaction prefixes are configurable and indicate whether:
• The item is Renewable (RE-)
• The item is Library Use Only (LU-)
• The item is Library Use Only and Renewable (LR-)
• The item is not Renewable and is not Library Use Only (TN-)

```lua
local tn = "";
if Settings.Use_Prefixes then
    local t = GetFieldValue("Transaction", "TransactionNumber");
    if GetFieldValue("Transaction", "LibraryUseOnly") and GetFieldValue("Transaction", "RenewalsAllowed") then
        tn = Settings.Prefix_for_LibraryUseOnly_and_RenewablesAllowed .. t;
    end
    if GetFieldValue("Transaction", "LibraryUseOnly") and GetFieldValue("Transaction", "RenewalsAllowed") ~= true then
        tn = Settings.Prefix_for_LibraryUseOnly .. t;
    end
    if GetFieldValue("Transaction", "RenewalsAllowed") and GetFieldValue("Transaction", "LibraryUseOnly") ~= true then
        tn = Settings.Prefix_for_RenewablesAllowed .. t;
    end
    if GetFieldValue("Transaction", "LibraryUseOnly") ~= true and GetFieldValue("Transaction", "RenewalsAllowed") ~= true then
        tn = Settings.acceptItem_Transaction_Prefix .. t;
    end
else
    tn = Settings.acceptItem_Transaction_Prefix .. GetFieldValue("Transaction", "TransactionNumber");
end
```

(Written in .lua)
ILLiad tells Aleph what to do via XML Message:
Borrowing: Configurable Transaction Prefixes

25 lines for the message

Easy Breakdown:
*From Unique Agency
*To Unique Agency
*Unique Agency ID
Item Identifier Value

In the **Borrowing workflow**, this item identifier could be ??-####### or TN-#######

In the **Lending workflow**, this item identifier could be the Transaction Number or Item Number

* Set in Addon Config
Borrowing: Configurable Transaction Prefixes

```plaintext
local tn = "";
if Settings.Use_Prefixes then
    local t = GetFieldValue("Transaction", "TransactionNumber");
    if GetFieldValue("Transaction", "LibraryUseOnly") and GetFieldValue("Transaction", "RenewalsAllowed") then
        tn = Settings.Prefix_for_LibraryUseOnly_and_RenewablesAllowed .. t;
    end
    if GetFieldValue("Transaction", "LibraryUseOnly") and GetFieldValue("Transaction", "RenewalsAllowed") ~= true then
        tn = Settings.Prefix_for_LibraryUseOnly .. t;
    end
    if GetFieldValue("Transaction", "RenewalsAllowed") and GetFieldValue("Transaction", "LibraryUseOnly") ~= true then
        tn = Settings.Prefix_for_RenewablesAllowed .. t;
    end
    if GetFieldValue("Transaction", "LibraryUseOnly") ~= true and GetFieldValue("Transaction", "RenewalsAllowed") ~= true then
        tn = Settings.acceptItem_Transaction_Prefix .. GetFieldValue("Transaction", "TransactionNumber");
    end
else
    tn = Settings.acceptItem_Transaction_Prefix .. GetFieldValue("Transaction", "TransactionNumber");
end
```

Lending: Configurable Transaction Field

```plaintext
local tn = "";
if Settings.checkInItem_UseItemNumber then
    tn = GetFieldValue("Transaction", "ItemNumber")
elseif Settings.checkInItem_UseTransactionNumber then
    tn = GetFieldValue("Transaction", "TransactionNumber");
else
    tn = GetFieldValue("Transaction", "TransactionNumber");
end
```
CheckInItem (Borrowing & Lending)

```java
local cib = '';
cib = cib .. '<NCIPMessage version="http://www.niso.org/ncip/v1_0/impl/dtd/ncip_v1_0.dtd">
  cib = cib .. '<CheckInItem>'
cib = cib .. '<InitiationHeader>'
cib = cib .. '<FromAgencyId>'
cib = cib .. '<UniqueAgencyId>'
cib = cib .. '<Scheme>' .. Settings.checkInItem_from_uniqueAgency_scheme .. '</Scheme>'
cib = cib .. '<Value>' .. Settings.checkInItem_from_uniqueAgency_value .. '</Value>'
cib = cib .. '</UniqueAgencyId>'
cib = cib .. '</FromAgencyId>'
cib = cib .. '<ToAgencyId>'
cib = cib .. '<UniqueAgencyId>'
cib = cib .. '<Scheme>' .. Settings.checkInItem_to_uniqueAgency_scheme .. '</Scheme>'
cib = cib .. '<Value>' .. Settings.checkInItem_to_uniqueAgency_value .. '</Value>'
cib = cib .. '</UniqueAgencyId>'
cib = cib .. '</ToAgencyId>'
cib = cib .. '</InitiationHeader>'
cib = cib .. '<UniqueItemId>'
cib = cib .. '<UniqueAgencyId>'
cib = cib .. '<Scheme>' .. Settings.checkInItem_uniqueItem_agency_scheme .. '</Scheme>'
cib = cib .. '<Value>' .. Settings.checkInItem_uniqueItem_agency_value .. '</Value>'
cib = cib .. '</UniqueAgencyId>'
cib = cib .. '<ItemIdentifierValue>' .. tn .. '</ItemIdentifierValue>'
cib = cib .. '</UniqueItemId>'
cib = cib .. '</CheckInItem>'
cib = cib .. '</NCIPMessage>'
return cib;
end
```
39 lines for the message

**Easy Breakdown:**
* From Unique Agency
* To Unique Agency
* Unique Agency ID
User Identifier Value (Pseudopatron Config)
Item Identifier Value (Reference Number)
* Unique Request ID
Request Identifier Value (Transaction Number)

* Set in Addon Config
Setting a Pseudopatron for Borrowers

- Do you want to use the OCLC Symbol for each Borrower?
- Do you want to prepend a string before the OCLC Symbol (e.g. “ILL-“)?
- Do you want to just use “ILL” to indicate the Borrower?
- Do you want to use any name in the world to indicate the Borrower?

```plaintext
function buildCheckOutItem()
local pseudopatron = "";
if Settings.Use_Lender_String == true then
    local OCLC_Sym = GetFieldValue("Transaction", "LendingLibrary");
    pseudopatron = Settings.Borrower_Prepend_String .. OCLC_Sym;
elseif Settings.Use_ILL_for_Library == true then
    pseudopatron = "ILL";
elseif Settings.BlanketName_for_Borrowing_Libraries ~= nil then
    pseudopatron = Settings.BlanketName_for_Borrowing_Libraries;
end
```
local coi = '';

coi = coi .. '<NCIPMessage version="http://www.niso.org/ncip/v1_0/impl/dtd/ncip_v1_0.dtd">
coi = coi .. '<InitiationHeader>'
coi = coi .. '<FromAgencyId>'
coi = coi .. '<UniqueAgencyId>'
coi = coi .. '<scheme>' .. Settings.checkOutItem_from_uniqueAgency_scheme .. '</scheme>'
coi = coi .. '<Value>' .. Settings.checkOutItem_from_uniqueAgency_value .. '</Value>'
coi = coi .. '</UniqueAgencyId>'
coi = coi .. '</FromAgencyId>'
coi = coi .. '<ToAgencyId>'
coi = coi .. '<UniqueAgencyId>'
coi = coi .. '<scheme>' .. Settings.checkOutItem_to_uniqueAgency_scheme .. '</scheme>'
coi = coi .. '<Value>' .. Settings.checkOutItem_to_uniqueAgency_value .. '</Value>'
coi = coi .. '</UniqueAgencyId>'
coi = coi .. '</ToAgencyId>'
coi = coi .. '<InitiationHeader>'
coi = coi .. '<UniqueUserld>'
coi = coi .. '<UniqueAgencyld>'
coi = coi .. '<scheme>' .. Settings.checkOutItem_uniqueUserAgency_scheme .. '</scheme>'
coi = coi .. '<Value>' .. Settings.checkOutItem_uniqueUserAgency_value .. '</Value>'
coi = coi .. '</UniqueAgencyId>'
coi = coi .. '<UserIdentifierValue>' .. pseudopatron .. '</UserIdentifierValue>'
coi = coi .. '</UniqueUserld>'
coi = coi .. '<UniqueItemld>'
coi = coi .. '<UniqueAgencyld>'
coi = coi .. '<scheme>' .. Settings.checkOutItem_uniqueItemAgency_scheme .. '</scheme>'
coi = coi .. '<Value>' .. Settings.checkOutItem_uniqueItemAgency_value .. '</Value>'
coi = coi .. '</UniqueAgencyId>'
coi = coi .. '<ItemIdentifierValue>' .. refnumber .. '</ItemIdentifierValue>'
coi = coi .. '</UniqueItemld>'
coi = coi .. '<UniqueRequestId>'
coi = coi .. '<UniqueAgencyld>'
coi = coi .. '<scheme>' .. Settings.checkOutItem_uniqueRequestAgency_scheme .. '</scheme>'
coi = coi .. '<Value>' .. Settings.checkOutItem_uniqueRequestAgency_value .. '</Value>'
coi = coi .. '</UniqueAgencyId>
coi = coi .. '<RequestIdentifierValue>' .. tn .. '</RequestIdentifierValue>'
coi = coi .. '</UniqueRequestId>'
coi = coi .. '</CheckOutItem>'
coi = coi .. '</NCIPMessage>'

return coi;
Making Changes to the Addon

Don’t use the Reset Cache button! Make changes directly to the code. Must close Client and re-open.

Don’t set anything under Manage Addons! These changes are only stored locally and sometimes don’t work!
function BorrowingAcceptItem(transactionProcessedEventArgs)
    luanet.load_assembly("System");
    local ncipAddress = Settings.NCIP_Responder_URL;
    local BAImessage = buildAcceptItem();
    local myWebClient = WebClient();
    myWebClient.Headers:Add("Content-Type", "text/html; charset=UTF-8");
    local BAIresponseArray = myWebClient:UploadString(ncipAddress, BAImessage);
    local currentTN = GetFieldValue("Transaction", "TransactionNumber");

    if string.find(BAIresponseArray, "Problem") then
        ExecuteCommand("Route", {currentTN, Settings.BorrowingAcceptItemFailQueue});
        ExecuteCommand("AddNote", {currentTN, LCIIresponseArray});
        SaveDataSource("Transaction");
    else
        ExecuteCommand("AddNote", {currentTN, "NCIP Response for BorrowingAcceptItem received successfully"});
        SaveDataSource("Transaction");
    end
end
function BorrowingAcceptItem(transactionProcessedEventArgs)
    LogDebug("BorrowingAcceptItem - start");

    luanet.load_assembly("System");
    local ncipAddress = Settings.NCIP_Responder_URL;
    local BAImessage = buildAcceptItem();
    LogDebug("creating BorrowingAcceptItem message[" .. BAImessage .. "]");
    local myWebClient = WebClient();
    LogDebug("WebClient Created");
    LogDebug("Adding Header");
    myWebClient.Headers:Add("Content-Type", "text/html; charset=UTF-8");
    LogDebug("Setting Upload String");
    local BAIresponseArray = myWebClient:UploadString(ncipAddress, BAImessage);
    LogDebug("Upload response was[" .. BAIresponseArray .. "]");

    LogDebug("Starting error catch")
    local currentTN = GetFieldValue("Transaction", "TransactionNumber");
4076285
*/
Problem Queues

Generic Problem Queues (Configurable Names):
• BorrowingAcceptItemFailQueue
• BorrowingCheckInItemFailQueue
• LendingCheckOutItemFailQueue
• LendingCheckInItemFailQueue

Growing List of Queues:
• BorrowingAccept: Item Not Checked Out
• BorrowingAccept: User Authentication Failed
• BorrowingAccept: Service is not known (unicode characters)
• BorrowingCheckIn: Unknown Item
• BorrowingCheckIn: Item Not Checked Out
• LendingCheckOut: No change in due date
• LendingCheckOut: User Ineligible To Check Out This Item
• LendingCheckOut: User Unknown
• LendingCheckIn: Unknown Item
• LendingCheckIn: Item Not Checked Out

This list may grow in the future (hopefully not)!
There are 7 pages with 57 errors!
The errors are not very useful!
Special Characters

Error! “Unknown Service” ???

Solution: Atlas Helpers URL encode:

Change this:

To this:

To this:
Assessment

• Beta period – June 2 – July 7
• Survey planned
• Feedback
  – Patrons
  – Staff
A few issues

• Aleph upgrade planned same weekend
  – Aleph v20 testing failed – security certificates
  – Aleph v22 upgrade failed – beta test delayed
• In-Library-Use Only workflow a little tricky
• A few bugs/tweaks along the way
  – Checking in articles created an item record in Aleph
  – Unicode
  – Returns would always result in an error because they are never charged out
• Lending implementation delayed
  – Waiting to establish patron database in Aleph