**Biligram 3.0**  | Installation guide

Daniel Rosenberg MD

Providence St Joseph Health, Portland OR, USA

October 2021

# Introduction

Biligram is a SMART on FHIR app designed to interface with electronic medical record software to plot pre-discharge newborn bilirubin values. It can help determine:

* Follow-up timing after discharge
* Suitability for phototherapy or exchange transfusion
* Trends in bilirubin values before, during and after treatment

As of version 3.0, Biligram can plot the following graphs:

* Bhutani risk levels;
* AAP 2004 phototherapy and transfusion therapy treatment thresholds for infants 35+ weeks gestational age at birth;
* NCNC (Northern California Neonatal Consortium) treatment thresholds for GA 35+ weeks
* Stanford Premie BiliRecs treatment thresholds for infants 37-35 weeks GA.

Biligram runs on a web server accessible by your EMR, using a SMART on FHIR interface to retrieve lab results and demographic information (including birth weight and gestational age), displays those results to clinicians, and allows for copying and pasting graphs into progress notes.

# Using Biligram

* Phototherapy will show up as vertical blue bands; hover for details.
* Patients without birth times (e.g., born outside the hospital) now prompt for a birth time.
* AAP 2004 treatment thresholds remain the default for term/early term (35w+) patients;
* The new Northern California Neonatal Consortium graphs, which are candidates to replace the 2004 AAP thresholds, are available as an option;
* For 27w – 34w6d GA patients, the graph will be based on Stanford’s Premie BiliRecs;
* For patients younger than 27 weeks, providers can view a Maisels threshold table.
* For infants who results cross a threshold only for infants with neurotoxicity risk factors, users will be prompted to indicate whether those risk factors are present (such as isoimmune disease, G6PD-deficiency, or sepsis). If this is toggled “on” then the “high risk” data points will also be highlighted (in addition to those crossing the “any infant” curve). In this fashion, we hope to reduce confusion: most infants without risk factors don’t qualify for treatment, even though they may have data points that cross a treatment threshold.

A picture containing chart

Description automatically generated

➊ Click on the “vertical 3 dots” menu for **Help pages**.

➋ Hover over the sidebar icons to **change the nomogram** (graph icon: choices include AAP, NCNC, Premie BiliRecs) or Neurotoxicity risk factors (exclamation point).

➌ Hover over data points or blue bands for specifics on timing, levels and phototherapy modalities.

➍ Click on items in the legend on the right to **change the visibility of treatment thresholds**.

➎ Click on the **Ñ** icon to dismiss a warning

When treatment thresholds are crossed, you’ll be prompted to **indicate whether an infant has neurotoxicity risk factors**, which will be listed. This helps highlight only the data points that may indicate treatment is needed:

Graphical user interface, text, application, email

Description automatically generated

Prompt to indicate whether a patient has neurotoxicity risk factors (same as under exclamation point icon; factors vary slightly between AAP and NCNC thresholds, as listed).

Graphical user interface, application

Description automatically generated

➏ The table shows follow-up recommendations based on the Bhutani nomogram. You can **toggle Hyperbilirubinemia risk factors**;

➐ The sidebar menu shows up if you hover over it; you can **toggle Neurotoxicity risk factors** (hover for a reference);

➑ **Copy graphs** by pressing the Copy button. The graph will be placed on the clipboard; you can then return to your progress note to paste it.

# Installation: what you’ll need

* A web server accessible by your EMR;
  + The web server should be whitelisted in your EMR;
  + Since protected health information (name, birthweight, gestational age, lab values) will be transmitted to this web server, it should be behind your firewall;
* A properly configured connection between your EMR and the web server where you’ll host Biligram (potentially together with other SMART on FHIR apps). We developed ours with Epic Interconnect;
  + This includes a Client ID and Client Secret; for Epic EMR users the Client ID is furnished from the [Epic App Orchard](https://apporchard.epic.com/Developer/Edit?appId=11284), and your Interconnect Administrator will generate Client Secrets for each of your servers (typically including a development, test, and production server).
* A development machine running **npm**, the Node.JS package manager, freely downloadable from <https://NodeJS.ORG>;
* The Biligram package, freely downloadable from <https://github.com/dmr001/biligram-3>
  + Biligram is free software, available under the [MIT License](https://opensource.org/licenses/MIT);
  + It uses the [Highcharts](https://shop.highsoft.com/) graphics library, which is freely available to non-profit organizations, but for-profit entities will need to pay a licensing fee to the publisher. (A suitable perpetual license is $535 as of 2021 for up to 20 developers and unlimited users.)

# Installation: initial setup

1. Download the Biligram package from <https://github.com/dmr001/biligram-3> to your development server.
2. Inspect the code. We can vouch for the safety of code contributed by the author (Daniel Rosenberg, github.com/dmr001) but forks from other authors should be treated carefully, as you will be giving this application read-only access to patient data.
3. You will need to add your client secrets and encoded versions of your phototherapy flowsheet row record ID’s to the package:

## Configure biligram/src/fhir/config.js

This file contains the SMART on FHIR client ID and client secrets, which give it permission to access your EMR. There are separate ID’s and secrets for your Production servers and staging/development (non-Production) servers. It also contains the encoded flowsheet row ID for the flowsheet row tracking phototherapy given to neonates.

Using Providence’s configuration as a template:

export default {  
 clientId: {  
 "apporchard.epic.com": "bdbb53cf-6d26-4cb0-a778-e04a94b22e99", // For Sandbox  
 "haikuor.providence.org": "bdbb53cf-6d26-4cb0-a778-e04a94b22e99", // For PRD  
 "haikuwa.providence.org": "bdbb53cf-6d26-4cb0-a778-e04a94b22e99", // For PRD  
 "haikuak.providence.org": "bdbb53cf-6d26-4cb0-a778-e04a94b22e99", // For PRD  
  
  
 "haikuor-staging.providence.org": "3d1fe54b-1efc-4858-8f2d-ebf549186abf", // Biligram Non-PRD  
 "haikuwa-staging.providence.org": "3d1fe54b-1efc-4858-8f2d-ebf549186abf", // Biligram Non-PRD  
 "haikuak-staging.providence.org": "3d1fe54b-1efc-4858-8f2d-ebf549186abf", // Biligram Non-PRD  
  
 "careeverywhereor-staging.providence.org": "3d1fe54b-1efc-4858-8f2d-ebf549186abf" // Biligram Non-PRD  
 },  
 clientSecrets: {  
 "apporchard.epic.com": "hIwCMavvb4t6z90BA42UOAdWvnzfhrGOlA4ntqhsxoPPSJpKm5ugMLYSxqhxoPPZXyYe203CISLsn/6Bm5EoSv==", // For Sandbox  
 "haikuor.providence.org": "hIwCMavvb4t6z90BA42UOAdWvnzfhrGOlA4ntqhsxoPPSJpKm5ugMLYSxqhxoPPZXyYe203CISLsn/6Bm5EoSv==", // For PRD  
 "haikuwa.providence.org": "hIwCMavvb4t6z90BA42UOAdWvnzfhrGOlA4ntqhsxoPPSJpKm5ugMLYSxqhxoPPZXyYe203CISLsn/6Bm5EoSv==", // For PRD  
 "haikuak.providence.org": "hIwCMavvb4t6z90BA42UOAdWvnzfhrGOlA4ntqhsxoPPSJpKm5ugMLYSxqhxoPPZXyYe203CISLsn/6Bm5EoSv==", // For PRD  
  
  
 "haikuor-staging.providence.org": "1AQH+O1A4ntqhsPPSJpK5ugMLYiiij8ak8V90a8IYMkYB0CzMrliOJ5EosvN4eCCeAxKKYAA/HXLhG47kVHlHG==", // Biligram - Non prod  
 "haikuwa-staging.providence.org": "1AQH+O1A4ntqhsPPSJpK5ugMLYiiij8ak8V90a8IYMkYB0CzMrliOJ5EosvN4eCCeAxKKYAA/HXLhG47kVHlHG==", // Biligram - Non prod  
 "haikuak-staging.providence.org": "1AQH+O1A4ntqhsPPSJpK5ugMLYiiij8ak8V90a8IYMkYB0CzMrliOJ5EosvN4eCCeAxKKYAA/HXLhG47kVHlHG==", // Biligram - Non prod  
  
  
 },  
 phototherapyFlowsheetCode: { // encoded phototherapy row for each host via .ZFHIRENDCODEFLO[306664127  
 // There's no other way to do this (as of 2021) except to get a unique encoded FLO row number for each  
 "haikuor.providence.org": 'http://open.epic.com/FHIR/STU3/StructureDefinition/observation-flowsheet-id|tDAY4zz2n-r4Iye9sQgXm0A0', // OCPRD  
 "haikuwa.providence.org": 'http://open.epic.com/FHIR/STU3/StructureDefinition/observation-flowsheet-id|tbBhRGwfHBAscSiy9E8N5kg0', // WMPRD  
 "haikuak.providence.org": 'http://open.epic.com/FHIR/STU3/StructureDefinition/observation-flowsheet-id|tUqd70DhzZN3cZQmyFjXb9A0', // AKPRD  
 "haikuor-staging.providence.org": 'http://open.epic.com/FHIR/STU3/StructureDefinition/observation-flowsheet-id|tPobsgZ1RwOFCIWGNeCQDIQ0', // OCTST  
 "haikuak-staging.providence.org": 'http://open.epic.com/FHIR/STU3/StructureDefinition/observation-flowsheet-id|t.LjUH46sreWjEFgWGGec-A0', // AKTST  
 "haikuwa-staging.providence.org": 'http://open.epic.com/FHIR/STU3/StructureDefinition/observation-flowsheet-id|t0wKUA7v4IHNf.snRjqlF2w0', // WMTST  
  
 }  
}

* The Client ID’s should remain unchanged, but you will need to change the names from our Interconnect servers (haikuOR, haikuWA, haikuAK, haikuOR-staging, haikuWA-staging, haikuAK-staging) to your own. If you only have a single staging and/or production server, delete the remaining ones.
* The Client Secrets are generated by your Interconnect analyst. THey should not be shared outside your organization. There should be one secret for your non-Production servers, and another for your Production server.
* You will need to encode the flowsheet row ID containing your phototherapy rows. For example, at Providence, we use FLO 306664127. (It would be nice to just query by the flowsheet row ID’s or a LOINC code, but our Epic doesn’t have this capability as of 2021). You can ask your DBA (database administrator) for the encoded row number, or (in Epic) use our code to make a SmartLink that will generate the encoded ID from the row number. We called ours ZFHIRENCODEFLOW; it runs this code:

s V(0)=1,V(1)=$$encodeFhirTokenFlowsheetRow^HULIBFHIRTOKEN(params)

Run that in each environment you’ll run Biligram in (testing and production) to get the encoded token for that environment. Following our example, replace our sample values in **biligram/src/fhir/config.js** with your own object keys and values, where the key is the hostname of the EMR server and the value is the encoded row token you came up with.

## Configure Biligram/src/fhir/FhirService.js

This file contains the [LOINC](https://loinc.org/) codes used to obtain lab and demographic values from your EMR. While we think you should not have to make many changes, they are worth checking in your setup:

|  |  |
| --- | --- |
| Birth weight | http://loinc.org|8339-4 |
| Gestational age at birth | http://loinc.org|76516-4 |
| Bilirubin values | http://loinc.org|1975-2, // Serum total bilirubin  http://loinc.org|58941-6, // Transcutaneous bilirubin http://loinc.org|42719-5, // Blood (POC) total bilirubin http://loinc.org|59828-4, // Blood (venous) total bilirubin http://loinc.org|35194-0 // Blood, total serum |

## Configure package.json

This file contains scripts used to copy Javascript code from your development machine to the web servers visible to your EMR. The “copy-to-cascade” line, for example, copies files over to our two load-balanced production servers; “deschutes” is our test server. Make your own version with the names of your staging and production servers and the appropriate pathnames.

{  
 "name": "biligram-3",  
 "version": "0.1.0",  
 "private": true,  
 "scripts": {  
 "mock": "json-server --watch ./src/mock/db.json",  
 "serve": "vue-cli-service serve",  
 "build": "vue-cli-service build",  
 "build-development": "vue-cli-service build --mode development",  
 "build-staging": "cross-env NODE\_ENV=staging vue-cli-service build --mode staging",  
 "lint": "vue-cli-service lint",  
 "copy-to-deschutes-old": "cd dist && copyfiles -u -V \*\*/\* //deschwebv12c1ff/d$/inetpub/wwwroot/EpicFtr/provonfhir/dmr/ && cd.. && echo 'copy complete'",  
 "copy-to-cascade-old": "cd dist && copyfiles -u -V \*\*/\* "copy-to-deschutes": "cd dist && copyfiles -u -V \*\*/\* //deschwebv12c1ff/d$/inetpub/wwwroot/EpicFtr/biligram-3.0 && cd.. && echo 'copy complete'",  
 "copy-to-cascade": "cd dist && copyfiles -u -V \*\*/\* //cascawebvde129f.wa.providence.org/d$/inetpub/wwwroot/EpicFtr/biligram-3.0 && copyfiles -u -V \*\*/\* //cascawebv0ac27c.wa.providence.org/d$/inetpub/wwwroot/EpicFtr/biligram-3.0 && cd.. && echo 'copy complete'",  
 "deploy-deschutes-old": "npm run build-staging && npm run copy-to-deschutes-old",  
 "deploy-deschutes-sandbox": "npm run build-staging && npm run copy-to-deschutes",  
 "deploy-deschutes-prov": "npm run build && npm run copy-to-deschutes",  
 "deploy-cascade": "npm run build && npm run copy-to-cascade"  
 },

## Configure transcutaneous bilirubin data entry

Note that, like many institutions, we initially stored our transcutaneous bilirubin values in a flowsheet row to ease data entry by support staff. But, that means you’d have to track bilirubin results both as lab values and flowsheet rows.

Currently, we have support staff use a flowsheet sidebar report which has a button for Biligram and another button for a version of the Results Console which allows end users to enter point-of-care bilirubin results.

Our flowsheet row then becomes read-only, displaying the result using the following extension:

1,%lrrNm(1)="BILIRUBIN",%extUser="1",x=$$GetLabCompNam^JFSPPT3

Our sidebar activity is configured with this HTML (in a SmartText record, **Biligram Flowsheet Sidebar buttons**):

<div style="background-color:white; font:arial; font-size:14pt;">

<a style="display:inline-block;padding:7px;border-radius:8px;margin:4px; color:white;background-color:#0046E7" href="EpicACT:OB\_BILIRUBIN\_CONSOLE\_ACTIVITY,RunParam:LaunchOptions:2">

Enter Results

</a>

<a style="display:inline-block;padding:7px;margin:4px;border-radius:8px;color:white;background-color:#93BC00" href="EpicACT:BILIGRAM\_AGL\_POPUP,RunParams:SOURCEID=2103420301">

Biligram

</a>

</div>

This, in turn, is display in a report, **IP Flowsheet docked report with Biligram**, which is a copy of Epic’s FLO 45345. You can duplicate that, and include a print group, **Biligram flowsheet sidebar buttons**. The is a printgroup that displays a SmartText, with the following code:

|  |  |
| --- | --- |
| 1 | s %ETXID=58519 d GetPntX^elibHULIB6(304100624) |

In this case, **58519** is the ETX shown above with HTML code, and **304100624** is an extension with the following code, all on one line:

n infoAry,etxGlo,go,etxLoop,saveGlobal,cerID,etxID,etxIN s saveGlobal=$S(^%ZeOSUNQ("mgr\_dir")["POC":0,^%ZeOSUNQ("mgr\_dir")["PROTO":0,^%ZeOSUNQ("mgr\_dir")["SANDBOX":0,^%ZeOSUNQ("mgr\_dir")["BLD":0,isdebug:0,1:%SAVEGLOBAL), cerID=%CERID, etxIN=%ETXID s %SAVEGLOBAL=0,%CERID="" s:%LUREPu="" %LSTREAM=1 s go=$S(cerID'="":$$evalRule^elibHULIB22(cerID,ID,DAT),1:1) i go=1 d setInfoAry^LUPPDB(.infoAry,"EPT",ID,DAT) f etxLoop=1:1:$L(etxIN,"^") s etxID=$P(etxIN,"^",etxLoop) s etxGlo=$$zGtTmpGlo^%Zefnlih(saveGlobal,"CACHEETX"\_etxID) s:@etxGlo@(0)="" %=$$SText^HSTLINK2(etxID,$S(%CONTEXT="":"1,800,35",1:%CONTEXT),"",ID,DAT,"","",1,"",.infoAry,etxGlo) x "n idx, line f idx=1:1:@etxGlo@(0) s line=@etxGlo@(idx) d setline^HUZLADD " d:+saveGlobal=0 %zRelTmpGlo^%Zefnlih(etxGlo)

Pulling that one line apart, here’s our documentation of the extension:

Use this to render and ETX as an "HTML" LPG. The LPG should be setup as HTML

Native and Standard, then this should be the code item.

s %ETXID=560721 d GetPntX^elibHULIB6(100624)

-or-

s %CERID=\*\*\*,%ETXID=\*\*\*,%CONTEXT="1,800" d GetPntX^elibHULIB6(100624)

PARAMS:

%ETXID - .1 of ETX to render. (may be caret delim list)

%CERID - .1 of a CER to evaluate and only render ETX if true (optional)

%CONTEXT - a comma-separated list of ETX-30 for functional types (optional,

when NOT provided "1,800" is used - regardless of setting in the ETX itself.

this is used for resolving smartlinks primarily)

%SAVEGLOBAL - cached the etx in a global and only load if it does not exist

there, this is great for things like STYLE, but does not evaluate any

smartlinks/rules again, so only useful for STATIC etx's not ones with "smarts"

built in

(never caches in POC, PROTO, or SANDBOX)

ASSUMES

ID

DAT

;local vars

n infoAry,etxGlo,go,etxLoop,saveGlobal,cerID,etxID,etxIN

s

saveGlobal=$S(^%ZeOSUNQ("mgr\_dir")["POC":0,^%ZeOSUNQ("mgr\_dir")["PROTO":0,^%ZeO

SUNQ("mgr\_dir")["SANDBOX":0,^%ZeOSUNQ("mgr\_dir")["BLD":0,isdebug:0,1:%SAVEGLOBA

L), cerID=%CERID, etxIN=%ETXID

;always reset the save global back to zero to not impact subsequent lpg's in

the lrp

s %SAVEGLOBAL=0,%CERID=""

;if this is a snapshot, then use "streaming" instead of buffering

s:%LUREPu="" %LSTREAM=1

;check the rule if it exists to determine if should be displayed

s go=$S(cerID'="":$$evalRule^elibHULIB22(cerID,ID,DAT),1:1)

i go=1

d setInfoAry^LUPPDB(.infoAry,"EPT",ID,DAT)

;loop over the list of ETX and render each

f etxLoop=1:1:$L(etxIN,"^")

s etxID=$P(etxIN,"^",etxLoop)

;allocate global

s etxGlo=$$zGtTmpGlo^%Zefnlih(saveGlobal,"CACHEETX"\_etxID)

;get the smart text into the global

s:@etxGlo@(0)=""

%=$$SText^HSTLINK2(etxID,$S(%CONTEXT="":"1,800,35",1:%CONTEXT),"",ID,DAT,"","",

1,"",.infoAry,etxGlo)

;iterate over the global and render each line

x "n idx, line

f idx=1:1:@etxGlo@(0)

s line=@etxGlo@(idx)

d setline^HUZLADD

"

;release global

d:+saveGlobal=0 %zRelTmpGlo^%Zefnlih(etxGlo)

Note this lets you do all sorts of amazing things anywhere a print group could be displayed (Biligram 2 used to run entirely inside one of these things) but safety concerns suggest you should carefully limit the use of these to approved and auditable HTML-containing SmartText records.

The Console activity referred to in the HTML (OB\_BILIRUBIN\_CONSOLE\_ACTIVITY) is a copy of other results console activities, with this set of initialization parameters:

Anchor=@PROGID=E\_MR\_MRVNANCHOR@,Inpatient=1,TEMPLATE=T\_OB\_BILIRUBIN\_RESULTS\_CONSOLE,TOC=0,STARTUPSECTION=SEC\_OB\_BILIRUBIN\_RESULTS\_CONSOLE

# Installation: standalone testing

Biligram comes with a built-in test patient you can use to make sur ethe graphing software works before you try to hook it up to your EMR. From a command line in the directory you downloaded the software into, you should be able to run:

npm install

Then, to start a local development server for your mock data, run the following:

npm run mock

And in another terminal window, run:

npm run serve

Bring up a web browser, and point it to the following address:

Localhost:8080/?mock=1

# Installation: EMR

We have been running Biligram on Epic version February 2021. We use a Web Integration (FDI) record, an Activity record (E2N) and a menu button record (E2U) to make it available to end users. You can also make a hyperlink to an activity record in suitable for inclusion in progress notes.

## Web integration record

In the Epic EMR, the web integration record will contain:

* Display name: **Biligram**
* Integration type: **SMART on FHIR**
* Client ID: The Client ID from the App Orchard (such as bdbb53cf-6d26-4cb0-a778-e04a94b22e99)
* Use actions queue for flowsheets: **No**
* Allow ordering by NDC ID: **No**
* Launch type (HTTPS only): **Workspace** (we actually use a floating window, but here it’s defined as Workspace)
* Authentication method: **SMART on FHIR**

## Activity record

You can make a copy of Epic’s CI\_ACTIVE\_GUIDELINES record (70120), with the following changes in your copy:

* Caption: **Biligram**
* Tooltip: **Biligram**
* Description: **Bilirubin nomogram tool**
* Activity icon: **BABY\_BUTTERNUT**
* Form style: **128** (popup window)

## Menu record

You will want a Menu record to go with the above Activity record.

* Menu type: **Item**
* Description: **Biligram**
* Caption: **Biligram**
* Toolbar tooltip: **Biligram**
* Image: **BABY\_BUTTERNUT**
* Activity descriptor: <Your activity record above>
* Run parameters: (configured on the next screen)
* URL by FDI: <Your Web Integration/FDI record>

You can add the menu record to places your users can find it. To make a hyperlink you could include in dot phrases or note templates, you could use something like this:

Reviewed bilirubin data in [Biligram](EpicACT:BILIGRAM_AGL_POPUP,RunParams:SOURCEID=2103420301):

Born at @DBLINK(ept,110)@ @DBLINK(ept,111)@, GA @DBLINK(ept,19408)@, birth weight @DBLINKWGT(ept,19401)@

@LABLAST(bilitot:10,bili:10)@

**Risk calculation**: \*\*\*

The hyperlink contains: EpicACT:<the name of your Activity record>,RunParams:SOURCEID=<the number of your FDI record>

## Test your installation

Run your version of the npm script that copies Biligram over to your staging web server:

npm run <your staging server script>

We found it helpful to run Epic’s Active Guidelines in debug mode on our test server. In an Epic Text window, edit the System profile (LPR 1) and use Home-F9 to search for the Text label “Debug,” and set the item Debug mode on (LPR item 17305) to **Yes**.

Get into the chart of a test patient with bilirubin values. Try pressing your Biligram button or using the hyperlink to get into the activity. If it doesn’t launch as expected, you can copy the URL from the debug page and paste it into a web browser that has access to your web server and see what errors are displayed.