1.0 Install the LINKER Software

The PHIX Anonymizer / Linker is an instantiation of the BioSense Linker Tomcat web application and required database tables, as well as the Mirth Connect channel destinations and transformers used to access the web application. The software and data involved are collectively referred to herein as the Linker. These notes detail the installation of the Linker. In addition, the usage of a Java class provided to test the Linker is described.

1.1 Installing the Linker Database

PostgreSQL was selected as the database software for use with PHIX. Installation of PostgreSQL 9.0, including the pgAdmin III tool, is described elsewhere in this document.

The PHIX Linker database backup file is located at db\BioSenseLinker.backup in PHIX_1_2_0_0.zip. This file is used to create the Linker tables, as follows:

1. Start the pgAdmin III tool.
2. Create a new, empty database called BioSenseLinker.
    a. In the Object browser, expand Server Groups -> Servers, and double-click the name of the desired server; enter a password if prompted.
    b. Under the desired server, right click Databases and select the New Database option.
    c. A pop-up appears. For database Name, enter BioSenseLinker. Select an Owner for the database from the list of possible owners. Make a note of the selected owner; this should be a database user for which the database password is known. Other fields should be okay to leave as they are. Click OK to close the pop-up and create the database.
3. Load the Linker database tables.
    a. In the Object browser, an entry appears for the BioSenseLinker database. Right click on this entry and select Restore… from the presented options.
    b. A pop-up appears. For the Filename field, click the … button to the right of the field and browse to the database backup file (see above for location info). Other fields should be okay to leave as they are. Click OK to load the backup file. The backup file should load relatively quickly. Click Done to close the pop-up.
    c. In the Object browser, expand BioSenseLinker -> Schemas -> public -> Tables, and verify that eight tables have loaded. These tables are called: tblassigningauthority, tblfacilityusers, tblpatientid, tblpatientidmergehistory, tblpatientname, tblvisitid, tblvisitidmergehistory, and tblvisitidref. All tables are empty except tblassigningauthority and tblfacilityusers.
4. Install appropriate assigning authorities.
    a. Right click on the tblassigningauthority table name and select View Data -> View All Rows.
    b. A pop-up appears, displaying two records with data in four fields:

1	LAB1	2.16.840.1.114222.4.3.2.5.2.6001	MRN

2 LAB1 2.16.840.1.114222.4.3.2.5.2.6001 AN

c. Double click in any field to edit it. Replace LAB1 by the human-readable name of the desired assigning authority. Replace 2.16.840.1.114222.4.3.2.5.2.6001 with the facility code for the desired assigning authority. MRN indicates that the record is for the assigning authority for Medical Record Numbers, and AN indicates that the record is for the assigning authority for Account Numbers.
d. Multiple assigning authorities may be listed in the database. Enter new records for other assigning authorities if this single Linker installation will be used for multiple facilities. The first field (linkerassigningauthority) in a record is a 1-up number uniquely identifying the record. For example, if a third record is entered, its first field would be 3.

5. (Optional) Install appropriate facility users. If this step is not performed, then only the Admin user will be able to access the database records. The Mirth Connect channel is configured to authenticate with the Admin user, so it will work without the need to install other facility users. The Admin user can be used with the Linker query form as well, so if there will be only one user of the Linker query form and that user should have access to all records, there is no need to install additional facility users. However, if it is desirable to allow non-Admin users to query the database with the Linker query form, these users must be installed as described here. Installing users in this table grants access to those users on a facility by facility basis. *Note: Users installed in this manner must still be created as Tomcat users, described below.*
    a. Right click on the tblfacilityusers table name and select View Data -> View All Rows.
    b. A pop-up appears, displaying three records with data in two fields:

        autoconsole 1326147224

        autoconsole 1881631646

        system 1447342373

    c. Double click in any field to edit it. Replace the first field with the name of an authorized user of the Linker. Replace the second field with the facility code for the facility for which the user is authorized to view data. The following example creates a user mike with access to records for a facility with facility code 2.16.840.1.114222.4.3.2.5.2.6001:

        mike 2.16.840.1.114222.4.3.2.5.2.6001

    d. Additional records may be created to install as many users as desired.

1.2 Installing Tomcat and the Linker Web Application

The Linker runs in a Java web container, and Tomcat 7.0 was chosen for this purpose in the PHIX implementation.

The Tomcat binaries and libraries and the Linker web application are located at dependencies\Linker\Tomcat in PHIX_1_2_0_0.zip. This folder is used to install Tomcat and the Linker web application as follows (a familiarity with XML terminology, such as tags and attributes, is assumed here):

1. Copy the entire folder named above to another location on disk, from which Tomcat will be run. This new location will be referred to as <TOMCAT_HOME>.
2. Ensure that <TOMCAT_HOME> and all subfolders are writable.
    a. In Windows Explorer, browse to the folder containing <TOMCAT_HOME>.
    b. Right click on the <TOMCAT_HOME> folder and select Properties.
    c. A pop-up appears. Near the bottom of the pop-up is an Attributes section. Ensure the Read-Only checkbox is not set and click Apply. If asked whether to apply the setting to all subfolders, respond in the affirmative.
3. (Optional) Install Tomcat as a Windows service. This step enables Tomcat to be run as a Windows service; the alternative is to run Tomcat via start and stop scripts, as described below.
    a. As an administrator user, open a Command Prompt and cd (change directory) to <TOMCAT_HOME>\bin.
    b. To install the service, enter this command: service.bat install
    c. To remove the service, enter this command: service.bat remove
    d. The service may then be administered as other Windows services; engage the assistance of a system administrator if necessary to start the service or to change the startup type to Automatic if it is desired to have Tomcat start automatically when the machine is booted.
4. (Optional) Create shortcuts to start and stop Tomcat manually, rather than as a service.
    a. In Windows Explorer, browse to <TOMCAT_HOME>\bin.
    b. Right click on the file startup.bat and select Create Shortcut. A shortcut item is created. Drag this shortcut item to the Desktop or wherever is desired.
    c. Right click on the file shutdown.bat and select Create Shortcut. A shortcut item is created. Drag this shortcut item to the Desktop or wherever is desired.
    d. Rename the shortcut items as desired. Use the startup shortcut to start Tomcat. This creates a Command Prompt window which you should not close via Windows controls. Use the shutdown shortcut to stop Tomcat.
5. Create Tomcat users. These Tomcat users include the Admin user and any other facilities users installed in the PostgreSQL tblfacilityusers table. (See the Linker database installation instructions above.) This process creates a double layer of security. First, a user must *authenticate* with Tomcat. Second, the user must be *authorized* to view data related to a given facility. The Linker database stores authorization data, and the web

application refers to the database to enforce authorization, whereas Tomcat enforces authentication, referring to some other data store to determine what users may be authenticated. The method in which Tomcat user data is created and stored can vary from one Tomcat installation to another, but here will be described the simplest method. Engage the assistance of a Tomcat expert to employ other means of creating Tomcat users, or to store passwords in an encrypted format.

a. In Windows Explorer, browse to <TOMCAT_HOME>\conf.
b. Open the file tomcat-users.xml in Notepad or some other text editor.
c. Scroll to the bottom of the file. Locate the lines that begin with the user tag. Change the username and password attributes in these lines to the user names and passwords desired. Make sure to include the Admin user. Otherwise, the user names should match the user names installed as facilities users in the Linker database.
    i. The passwords provided are not encrypted. It is possible to enter encrypted passwords, but what is necessary for this is not detailed here.
    ii. Users and roles listed in the tomcat-users.xml file may be removed if not needed. To use the Tomcat manager application, however, you need to include the manager-gui role and have that role assigned to any user who will make use of the manager application; there is an example of this for the user mike in the provided users file. It is necessary for the bioSense role to be included and to be assigned to the Admin user and to each facility user.
    iii. The password given to the Admin user here must be made known to the Mirth Connect channel that accesses the Linker web application. As provided, the Mirth Connect channel and the Linker Java test application are hardcoded to use the Admin password in the provided tomcat-users.xml file. A change to the Mirth Connect channel (see below), and to the Linker Java test application if it will be used (see below), is necessary if the Admin password is changed from that provided in the tomcat-users.xml file.

6. Customize the ports used by Tomcat. The provided Tomcat configuration uses ports 8005, 8009, 8444, and 8500. If any of these ports conflict with the ports used by other applications, the ports used by Tomcat may be changed.
    a. In Windows Explorer, browse to <TOMCAT_HOME>\conf.
    b. Open the file server.xml in Notepad or some other text editor.
    c. Scroll through the file and look for uncommented lines with port or redirectPort attributes. Change these values to those desired.
    d. The provided configuration sets port 8500 for the HTTP/1.1 protocol. It is only the HTTP/1.1 protocol port that is used elsewhere by the Linker. Other port values may be changed to any suitable values without requiring changes

elsewhere in the Linker. If the port for the HTTP/1.1 protocol is changed, then the Mirth Connect channel must be changed to match the HTTP/1.1 port used by Tomcat, as described in the Mirth Connect channel installation instructions below. The Linker Java test application can be passed an argument to instruct it to use a different HTTP port (in the baseUrl argument, described below), though it defaults to port 8500.

7. Change the user name and password for the Linker web application's connection to the Linker database. This is necessary if the user name and password for the Linker database is not the same as in the provided Hibernate configuration file for the Linker web application.
    a. In Windows Explorer, browse to <TOMCAT_HOME>\webapps\bioSense\WEB-INF\classes.
    b. Open the file hibernate.cfg.xml in Notepad or some other text editor.
    c. Near the top of the file, look for the uncommented line with the property tag and the connection.username attribute. Change the content of this tag to the required database user name.
    d. A similar line will have the connection.password attribute. Change the content of this tag to the required database password.
    e. Note that if a different database than PostgreSQL is used, a different hibernate.cfg.xml file will be required. There are example files for other database types in the same folder, but these files have not been tested by the PHIX team. The use of another database type will require the appropriate database driver to be installed; such customization is not described here.
8. (Optional) Customize other Tomcat or Linker web application configuration settings if desired or necessary. A Tomcat expert can make other customizations, such as defining a different URL for the Linker web application, different user roles, etc. Such advanced options are beyond the scope of this document.

1.3 Installing the Mirth Connect Channel for testing (optional)

An exported Mirth Connect channel is provided for Mirth 2.1.0, and is set up to exercise the Linker by reading input files from a particular folder, then writing output to the Mirth Connect log and to files on disk in another folder. If using Mirth 2.1.0, the channel definition may be imported from mirth_channels\development\LinkerTest.xml in PHIX_1_2_0_0.zip. Note that this is a different version of Mirth than the PHIX currently uses. The Import Channel command is located in the Channel Tasks area in Mirth Connect, and is straightforward to use in Mirth Connect 2.1.0. This exported channel has been integrated into the larger PHIX Mirth application in Mirth 2.0.1, but rather than attempting to extract the required Mirth Destinations and Transformers from there, if using an earlier version of Mirth Connect than 2.1.0, or if the Mirth Connect Linker channel is to be integrated with an existing Mirth Connect channel, a Mirth developer can install the Linker channel Destinations as follows (familiarity with Mirth Connect is assumed):

1. Start Mirth Connect Administrator.
2. Edit the channel into which the Linker is to be integrated, or create a new channel. The provided Mirth 2.1.0 Connect channel defines a Source that reads files from a folder, but this is not suitable for all purposes, so is not detailed here. The Source for the channel should result in reading in HL7 ADT data. For testing purposes it is fairly straightforward to create a FileReader Source as in the provided exported Mirth Connect 2.1.0 channel, so that HL7 ADT data files can be dropped into a specified folder and read from there by the Mirth Channel.
3. Create two Destinations, one named Linker Destination, the other named Linker Output. (These names are arbitrary, but if the first name is changed from Linker Destination, a similar name change must be made inside the transformer for the second Destination, as will be described in the steps below.)
    a. Create the Linker Destination. On the Destinations tab for the Mirth Channel, select New Destination from the Channel Tasks command area. A new Destination is created in the list of Destinations. Double click on the name of the newly created Destination in the Destinations list and rename the Destination to Linker Destination.
        i. Set the Connector Type to HTTP Sender.
        ii. The configuration area displays the fields for configuring a Destination of type HTTP Sender. Fill in the fields according to the following. Note that values followed by (?) may need to be set differently, depending on the system on which the Linker runs, and such possible settings are not detailed here. If the Admin Tomcat user was given a different password than that in the provided tomcat-users.xml file (see Tomcat installation instructions above), it is necessary to use that same password here. The URL used must match that of the Linker web application; for a discussion on how to determine this, see the instructions for running the Linker test application below.

            URL: http://localhost:8500/bioSense/Identifiers (?)

            Method: GET

            Send Timeout (ms): 30000 (?)

            Send Response to: None (?)

            Include Response Headers: No (?)

            Authentication: Yes

            Authentication Type: Basic

            Username: Admin

            Password: Admin (?)

            Use Persistent Queues: No (?)

Charset Encoding: UTF-8 (?)

Query Parameters (set up the below-described transformer first to enable dragging of variables to the Value column):

dsPatientAssigningAuthority: ${patientAssigningAuthority}

dsPatientIdType: ${patientIDType}

middleName: ${patientMiddle}

dsVisitIdType: ${visitIDType}

dsPatientId: ${patientID}

messageType: ${msgType}

dsVisitAssigningAuthority: ${visitAssigningAuthority}

dsVisitId: ${visitID}

firstName: ${patientFName}

lastName: ${patientLName}

b. Create the transformer for the Linker Destination.
   i. With Linker Destination selected in the Destinations list, click on Edit Transformer in the Channel Tasks command area.
   ii. The Transformers list appears. Double click on the name of the new Transformer and rename it as desired. To the right of the Transformer name is a Type pulldown; from the pulldown, select JavaScript.
   iii. A JavaScript editing area appears below the list of Transformers. In the editing area, enter the following script:

```
SerializerFactory.getHL7Serializer().toXML(message);

logger.info("Setting msgType=" +
msg['MSH']['MSH.9']['MSH.9.2'].toString() );
channelMap.put('msgType', msg['MSH']['MSH.9']['MSH.9.2'].toString() );

logger.info("Setting patientFName=" +
msg['PID']['PID.5']['PID.5.2'].toString() );
channelMap.put('patientFName', msg['PID']['PID.5']['PID.5.2'].toString()
);

logger.info("Setting patientLName=" +
msg['PID']['PID.5']['PID.5.1'].toString() );
```

```
channelMap.put('patientLName', msg['PID']['PID.5']['PID.5.1'].toString()
);


logger.info("Setting patientMiddle=" +
msg['PID']['PID.5']['PID.5.3'].toString() );

channelMap.put('patientMiddle', msg['PID']['PID.5']['PID.5.3'].toString()
);


logger.info("Setting patientID=" +
msg['PID']['PID.3']['PID.3.1'].toString() );

channelMap.put('patientID', msg['PID']['PID.3']['PID.3.1'].toString() );


logger.info("Setting patientIDType=MRN");

channelMap.put('patientIDType', "MRN");


logger.info("Setting patientAssigningAuthority="
        +msg['MSH']['MSH.4']['MSH.4.2'].toString() );

channelMap.put('patientAssigningAuthority',
        msg['MSH']['MSH.4']['MSH.4.2'].toString());


logger.info("Setting visitID=" +
msg['PV1']['PV1.19']['PV1.19.1'].toString() );

channelMap.put('visitID', msg['PV1']['PV1.19']['PV1.19.1'].toString() );


logger.info("Setting visitIDType=AN");

channelMap.put('visitIDType', "AN");


logger.info("Setting visitAssigningAuthority=" +
        msg['MSH']['MSH.4']['MSH.4.2'].toString() );

channelMap.put('visitAssigningAuthority',
        msg['MSH']['MSH.4']['MSH.4.2'].toString() );
```

iv. Click the Back to Channel command in the Mirth Views command area.

c. Create the Linker Output destination. On the Destinations tab for the Mirth Channel, select New Destination from the Channel Tasks command area. A new Destination is created in the list of Destinations. Double click on the name of the newly created Destination in the Destinations list and rename the Destination to

Linker Output. (Another name may be used if desired without requiring any changes to the provided Linker configuration.)

i. Set the Connector Type according to the desired Destination type; this could be a File Writer for testing purposes, as is done in the provided exported Mirth Connect channel. The configuration for the Connector Type for this Destination depends on the intended use of the Linker data and thus is site specific and not described here.
ii. Note that after the transformer (see below) is created for this Destination, there will be two variables available for use in this Destination: ${BioSensePatientId} and ${BioSenseVisitId}.

d. Create the transformer for the Linker Output destination.
   i. With Linker Output selected in the Destinations list, click on Edit Transformer in the Channel Tasks command area.
   ii. The Transformers list appears. Double click on the name of the new Transformer and rename it as desired. To the right of the Transformer name is a Type pulldown; from the pulldown, select JavaScript.
   iii. A JavaScript editing area appears below the list of Transformers. In the editing area, enter the following script (note: If the first Mirth channel Destination was called something other than Linker Destination, substitute the name used for the first Destination wherever Linker Destination occurs in this code):

```
var resp = new XML(responseMap.get('Linker
Destination').getMessage());

logger.info("inside transformer, processing Linker Destination response");
logger.info(resp);

var xml = new XML(resp);

logger.info("BioSensePatientId=" + xml['sBioSensePatientId']);
logger.info("BioSenseVisitId=" + xml['sBioSenseVisitId']);

channelMap.put('BioSensePatientId', xml['sBioSensePatientId']);
channelMap.put('BioSenseVisitId', xml['sBioSenseVisitId']);
```

   iv. Click the Back to Channel command in the Mirth Views command area.

4. When finished, click Save Changes in the Channel Tasks command area. If any errors occur, they must be addressed before the channel can be used.

1.4 Testing the Tomcat Linker Web Application (optional)

A Java test application is provided as part of the Linker installation package, and can be found at dependencies\Linker\src\LinkerTester in PHIX_1_2_0_0.zip. Extract this folder to a location on disk; this extracted folder will be referred to as <TESTER>. Both the source for the tester and the compiled classes are provided, so it can be used as is or modified and rebuilt by a Java developer. The folder is set up as an Eclipse project for the convenience of any Java developer assigned to modify the tester. It may be desirable to copy the entire folder to the developer's workspace and ensure that the folder and all subfolders are writable before attempting any changes.

To use the tester as provided, the PostgreSQL database server, Tomcat server, and Linker web application must be running. Mirth Connect need not be running, as the tester virtually takes the role of Mirth Connect. The bin folder for the tester is <TESTER>\bin, and either it must be included in the class path for Java, or Java must be executed from that folder. The fully qualified name of the class must be specified when executed. In the example usage below, the case of executing Java from <TESTER>\bin will be examined.

The tester should be run with Java 1.6, the same version of Java used with Tomcat. The tester need not be running on the same machine as Tomcat, however, as long as network access to the Tomcat machine is granted on port 8500 (or whatever other port Tomcat is configured for using the HTTP/1.1 protocol).

The URL for the Tomcat Linker web application must be determined, and is of a format that uses the same parts described above for querying Linker data via the web browser. Namely, <DOMAIN>, <PORT>, and <APP> are determined for the tester in the same way they are determined for the web browser. The URL to use with the tester is of the following form:

http://<DOMAIN>:<PORT>/<APP>/Identifiers

For instance, if the tester is run on the same machine as Tomcat and the <PORT> and <APP> values are taken as provided in the Tomcat configuration, the appropriate URL to use with the tester is as follows:

http://localhost:8500/bioSense/Identifiers

The tester as provided accepts up to six arguments. These arguments are, in the order expected on the command line: baseUrl, msgType, patientId, visitId, lastName, and firstName. Any arguments may be omitted, but omitting one means you must omit all that follow it. The empty string may also be used for any argument. If an argument is missing or the empty string, a default value is used.

- baseUrl: The URL of the Linker web application, as described above, ending with the Identifiers qualifier. Defaults to http://localhost:8500/bioSense/Identifiers.
- msgType: The type of ADT message involved, such as A01, A04, etc. Defaults to A01. Note that the assumption is that the message involved is an ADT message. (Since the message itself is not sent to the Linker, there is no way for the Linker to verify that the message is an ADT message.)

- patientId: The MRN of the patient, also referred to as the Hospital Patient Id in the Linker query form. Defaults to 1143302 for test purposes.
- visitId: The AN for the patient's visit, also referred to as theHospital Visit Id in the Linker query form. Defaults to 20524286 for test purposes.
- lastName: The patient's last name. Defaults to Newman for test purposes.
- firstName: The patient's first name. Defaults to Alfred for test purposes.

Six pieces of information required by the tester are not passed as arguments, but are hard-coded in the Java code. These are the Admin user name and password, the ids of the assigning authorities for patient and visit ids, and the patient and visit id types. The Admin user name and password are set to the Admin user name and password as set up in the provided Tomcat configuration. The assigning authority ids for patients and visits are both set to 2.16.840.1.114222.4.3.2.5.2.6001. The patient and visit id types are set to MRN and AN, respectively. For testing purposes, these should suffice, with the exception that if the Admin password is changed in Tomcat, it must also be changed in the tester, and the tester Java class recompiled. Other hard-coded values may be necessary to change for a given situation, and any Java developer should be qualified to make such changes, even to the extent of making it possible to pass in any of these hard-coded values as arguments, possibly even completely redefining the manner in which arguments are passed in.

To run the tester with default values for all arguments, open a Command Prompt window, cd (change directory) to <TESTER>\bin, and run the following command:

java com.saic.phlissa.linker.LinkerTester

To change the default value for an argument, include that argument on the command line. If the argument comes after other arguments that you wish not to change from the default, enter a pair of double quotes for those arguments you wish to default. For example, if you want to keep the default baseUrl argument, but change the other arguments, you could run the following command:

java com.saic.phlissa.linker.LinkerTester "" A04 1536 2077 Doe John

This example command would use the default URL for the Linker web application, pass A04 for the ADT message type, 1536 for the patient id, 2077 for the visit id, Doe for the patient's last name, and John for the patient's first name.

For a brief help message that shows the arguments and their expected order, run the command with a first argument of –h or ?, as follows:

java com.saic.phlissa.linker.LinkerTester -h

After running the tester, use the Linker Query page in the web browser or a database admin tool to verify the data was entered in the database.

In general, the output of the tester is the XML received from the Linker web application, which can be parsed to obtain the generated, anonymized patient id and visit id. The string ***** Done ***** is appended to the output, but if this is not desired, a Java developer can easily remove it and recompile the code.

1.5 Modifying the Source Code for the Linker Web Application

The source code for the Linker web application is in dependencies\Linker\src\Eclipse in PHIX_1_2_0_0.zip, where it is set up as an Eclipse project. A Java developer who wishes to modify the source code is recommended to copy the entire folder to their workspace and ensure that the folder and all subfolders are write-enabled. The project may then be edited as any standard Eclipse Java project.

2.0 Running the Linker

Running the Linker requires the PostgreSQL database server, the Tomcat server, and the Mirth Connect server to be running with the installed database, web application, and Mirth Connect channel as previously described. Typically, these servers can be run automatically on machine startup by installing them as Windows services. The installation of Tomcat as a Windows service was described above. The installation of PostgreSQL and Mirth Connect as services is typically done as part of the installation of those programs. These installations are described elsewhere in this document.

When all three servers are running, HL7 ADT data can be fed to the Linker according to the requirements of the Source for the Linker channel in Mirth Connect. The data flow is as shown here.

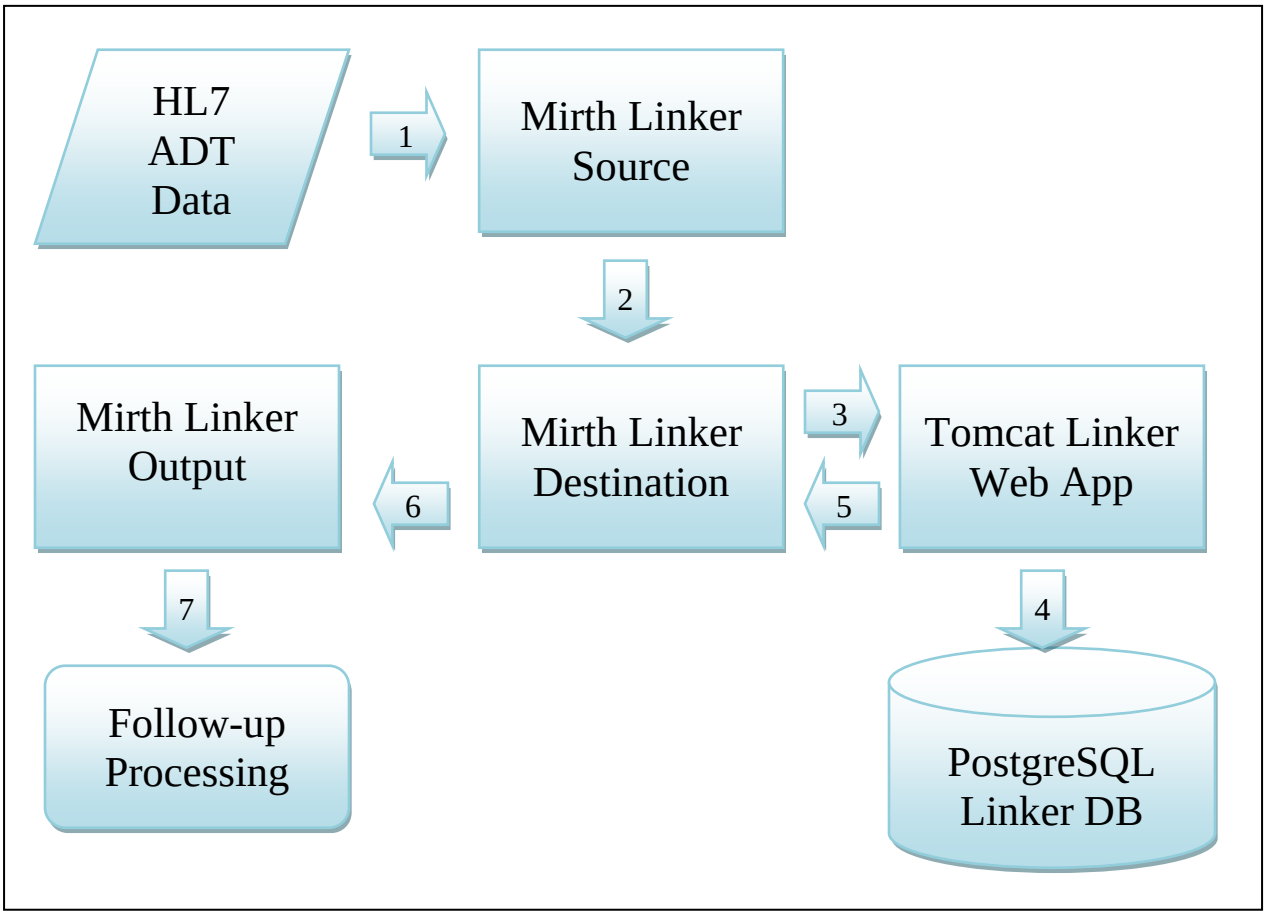

Figure 1. Data Flow and LINKER Integration

As depicted, 1) HL7 ADT data is read by the Mirth channel as defined by the Mirth Linker Source. It is then 2) passed to the Mirth Linker Destination, which extracts HL7 fields and constructs the URL for 3) connecting to the Tomcat Linker web application. The web application authenticates Mirth as the Admin user, then generates anonymous patient and visit ids and 4) updates the PostgreSQL Linker database with the patient and visit info as necessary, based on the URL query parameters. The web application 5) returns the generated patient and visit ids to the Mirth Linker Destination in a standard HTTP response, consisting of XML formatted data. The

response is 6) read and parsed by the Mirth Linker Output, which extracts the anonymized patient and visit ids. The Mirth Linker Output is then responsible for 7) sending anonymized patient and visit ids to a follow-up process or writing the anonymized ids to storage for easy access later in querying the web application for the original patient and visit information. Follow-up processing should retain facility codes and associate them with generated patient and visit ids.

2.1 Querying the Linker for Original Patient and Visit Information

Making a query for the original patient and visit information requires at a minimum that the PostgreSQL database server be running. Anyone with the PostgreSQL user name and password may connect to the database and run standard SQL queries or use a GUI admin tool to view or query the database. This, however, is not the preferred way to view the data, as there is no intent for the tables to be easily interpreted by human eyes. The Tomcat Linker web application includes two JSP files, one for querying and the other for viewing Linker data through the use of a standard web browser, displaying results in a form more suitable to human viewing. Only the URL of the query JSP page need be directly entered in the web browser, as described in this section.

To use the Linker query JSP page from a web browser, Tomcat and the Linker web application must be running. The web browser need not be running on the same machine as Tomcat, but must have network access to the Tomcat machine. For purposes of this discussion, the Tomcat machine domain name will be referred to as <DOMAIN>. Tomcat will also have been configured to use a particular port for the HTTP/1.1 protocol, and this port will be referred to as <PORT>. This port is configured as 8500 in the provided installation files. Lastly, the Tomcat Linker web application will have been configured to run in a particular subfolder of <TOMCAT_HOME>\webapps. This subfolder will be referred to as <APP>; in the provided installation files, this subfolder is bioSense.

Given the above, the URL to enter into the browser's address bar is of the following form:

http://<DOMAIN>:<PORT>/<APP>/viewer/Query.jsp

For example, suppose the web browser is running on the same machine as Tomcat, allowing localhost to be used for <DOMAIN>. Given the provided defaults for <PORT> and <APP>, the URL would be as follows:

http://localhost:8500/bioSense/viewer/Query.jsp

The first time in each browser session that the correct URL for the Linker web application query page is visited in the browser, the browser will prompt for a user name and password. The Admin user and password may be used to access information for all patients in the database, or a facility user and password can be used to restrict access to only the patient information for which the facility user is authorized (see the Tomcat and PostgreSQL database installation instructions regarding the set up of facility users).

Once authorization is performed, the query page loads an HTML form which the user may fill in with appropriate data to retrieve original and anonymized patient and visit id numbers. Given any one such value, the web application can retrieve and display the other values associated with the particular patient or visit. Original ids are referred to as "Hospital" ids; generated, anonymized ids are referred to as "PHIX" ids. See Figure 1 for the query form. See Figure 2 for sample results of a query.

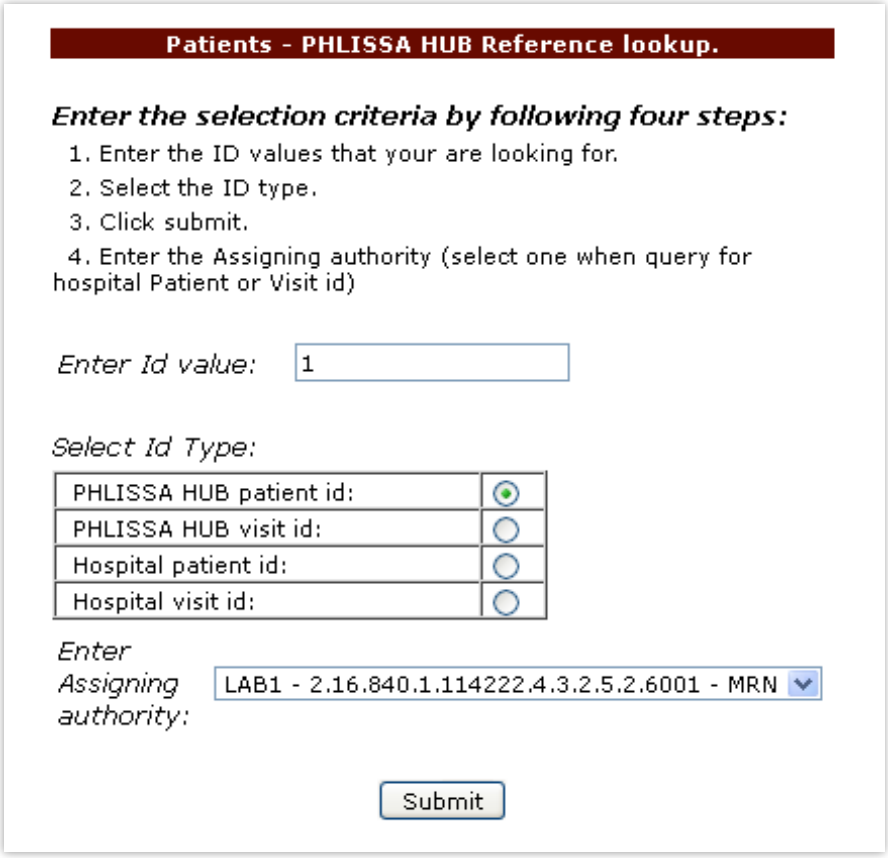

Figure 2: PHIX Patient and Visit Id Lookup Form

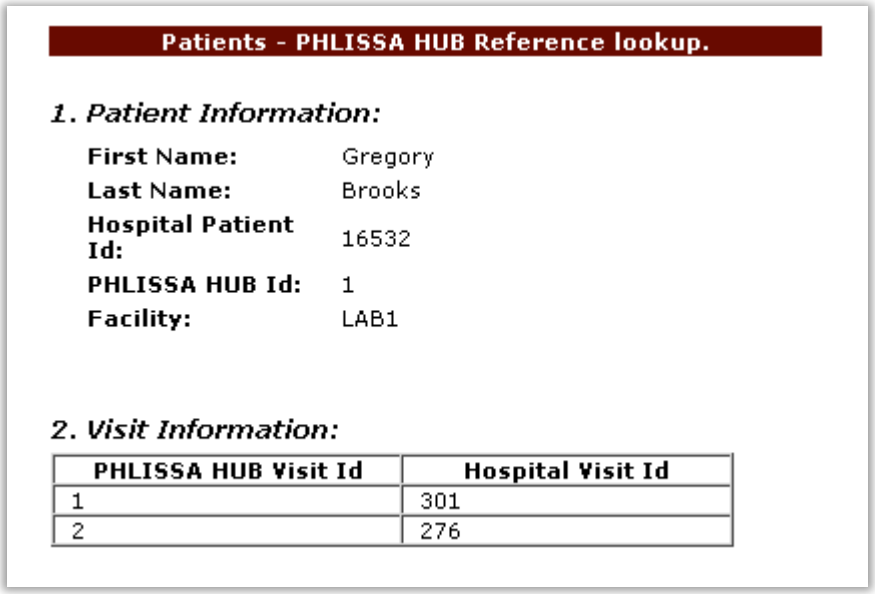

Figure 3: PHIX Patient and Visit Id Results Page

Note that the query form will accept any of four different types of ids: PHIX patient or visit id, or Hospital patient or visit id. Note that each generated PHIX patient id is unique within the database, and each generated PHIX visit id is unique within the database (i.e., no two generated patient ids are the same, and no two generated visit ids are the same—within the same Linker database). However, if more than one facility is tracked in the database, there is no guarantee that the Hospital patient id for one patient at a given facility is not the same as the Hospital patient id for another patient at another facility. The same may be said for Hospital visit ids. Thus, when the query is based on a Hospital id, the desired facility and account type must be selected from the assigning authority pulldown menu, an example of which may be seen in Figure 1. Selecting from the assigning authority pulldown menu is not required when entering a PHIX patient or visit id.

As shown in Figure 1, any query made is a simple one based on a single id of a single type. For more complicated queries, the admin tool or other database query tool must be used by a database expert.

When logged in as a facility user (as opposed to the Admin user), the assigning authorities pulldown menu will only contain those facilities for which the facility user is authorized. Moreover, for a facility user, results will only be returned for patients and visits associated with the facility for which the user is authorized. For instance, if a facility user is authorized access to one facility and not the other, any patient or visit data returned for view by the facility user will include data associated with the first facility and exclude any data associated with the second facility. For instance, if a patient visited both facilities, only the patient information and visit information for the first facility would be displayed. In this scenario, the patient would have two different generated PHIX patient ids, one for each facility visited. A query using one of the patient ids would only display results corresponding to the patient at the corresponding facility.

It should also be noted that for two different Linker installations, the generated patient and visit ids will not be unique across the installations. Generated patient id 1 in one Linker installation could be for a different actual patient than generated patient id 1 in another Linker installation. It is imperative to know which installation to query, which is why it is necessary for follow-on processing to associate facility codes with generated PHIX patient and visit ids, and why information dealing with any one given facility code should be used with only one Linker installation, regardless of how many facilities are tracked by the same installation. In any case, if a facility is presented with a generated PHIX patient id in connection with a particular facility code (which the facility must verify is indeed their facility code), the facility must know which Linker installation to query to find the original Hospital patient id. This is easy if the facility only makes use of one Linker installation, regardless of how many other facilities might make use of the same Linker installation.

If it becomes necessary for a single facility to use more than one Linker installation, the facility may need to make use of multiple facility codes, each of which is unique across Linker installations, but this is an implementation detail that is not discussed here.

In a practical example, suppose a Hospital must send info to a Public Health Lab (PHL). The Hospital would use a process involving the Linker to send the info to the PHL, and as part of this process, the Linker would generate PHIX patient and visit ids for sending to the PHL instead of the original Hospital patient and visit ids. Along with the generated ids, the Hospital would send the facility code of the Hospital. (Exactly what is sent is outside the realm of the Linker, but

anonymization would typically involve the replacement in an HL7 message of the original Hospital patient and visit ids with the generated ids, then forwarding the modified HL7 message to the PHL.) The PHL would record the received data in such a way that when a generated id is retrieved, so is the matching facility code. If the PHL then saw a need for some action to occur involving a particular patient, the PHL would use the facility code to determine which facility could identify the patient, in this case, the Hospital. The PHL would send the facility code and generated patient and/or visit ids to the Hospital, indicating the type of id(s) sent, patient and/or visit. The Hospital would verify that the facility code is indeed for the Hospital, and then use the Linker viewer for the correct Linker installation to run a query on the id(s) sent to the Hospital from the PHL. The viewer would return the patient information for the id(s), and the Hospital could then take action regarding the identified patient as requested by the PHL.