

# Interfacing with Databases in Rhapsody

# Welcome

- Database Communication Points
  - Database, database insertion
- Database Filters
  - Database lookup, database message extraction, generic code translation, HL7 external code translation, HL7 external code validation
- Configuration Editor
  - Connection details
  - Database drivers
  - SQL statement editor
- Stored Procedure Demo

# Database Communication Points

- **Database Communication Point**
  - **Input**-query database to read records which have been inserted or updated since the last time the query was run
  - **Output**-communication point executes the query in its configuration
  - **Out->In**-operates in a manner similar to the Database Lookup filter
  - **In->Out**-waits before sending out a message until a response to the previous message has been received



**Database**

# Sequence of Events in Input Mode

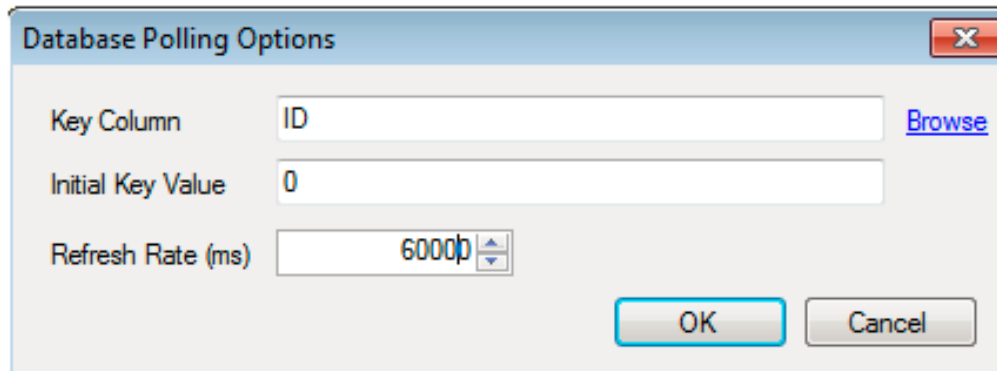
- The sequence of events for a Database communication point in Input mode is as follows:
  1. Query the database
  2. Records are returned
  3. Place records into a XML message and place on the route
  4. Immediately query the database again
    - If records are returned, repeat the sequence
    - If no records are returned, sleep for the period defined in the Refresh Rate parameter of the configuration and then query the database again.

# Database Communication point properties

- **Database**
  - The type of database with which the communication point is connecting
- **Configuration file**
  - XML file that controls how the communication point interacts with the database
- **Query Timeout**
  - Used to specify how long in seconds a remote operation can take before the procedure times out
- **Message Type**
  - Identifies the output message type when no definition is specified
- **Definition**
  - Identifies the message definition used to parse the message

# Polling Parameters

- Polling Parameters must be configured if the database communication point is in an Input mode
  - **Refresh Rate**-Controls the length of time that the communication point will sleep after the statements return no records
  - **Key Column**-Name of the field which must be returned by the query
  - **Initial Value**-Value used the first time the database is polled by the component



The screenshot shows a dialog box titled "Database Polling Options" with a close button (X) in the top right corner. It contains three input fields: "Key Column" with the value "ID" and a "Browse" button to its right; "Initial Key Value" with the value "0"; and "Refresh Rate (ms)" with a spinner box set to "6000". At the bottom right, there are "OK" and "Cancel" buttons.

# Database Communication Point Demo (Output/Input)

# Database Communication Points

- Database Insertion Communication Point
  - Output only
  - Optimized to insert or update records in a single database table
  - Used when the extra complexity of the database communication point is not required



Database  
Insertion



# Database Insertion Communication Point Properties

- **Definition**
  - Message definition that identifies the structure and type of incoming messages
- **Database**
  - Connection details for the database
- **JDBC Driver/Database URL**
- **Database Table**
- **Column Values**
  - Identifies columns to be updated
- **Mode**
  - Insert or update
- **Update Where Clause**
  - Identifies conditions under which the information in the database is to be updated

# Column Values

- Column Name
  - Name of the Column in the database
- Value
  - Value to be saved in the database from the HL7 message
- Type
  - Null
  - Body
  - Timestamp
  - Literal
  - Field
  - Property

	Column Name	Value	...	Type
1	FirstName	PID.PatientName[0].GivenName	...	Field ▼
2	FamilyName	PID.PatientName[0].FamilyName	...	Field ▼
3	InternalID	PID.InternalPatientID[0].IDNumber	...	Field ▼
4	ExternalID	PID.ExternalPatientID.IDNumber	...	Field ▼
5	AlternativeID	LocalID	...	Literal ▼
6	IssuingAuthority	IssuingAuthority	...	Property ▼

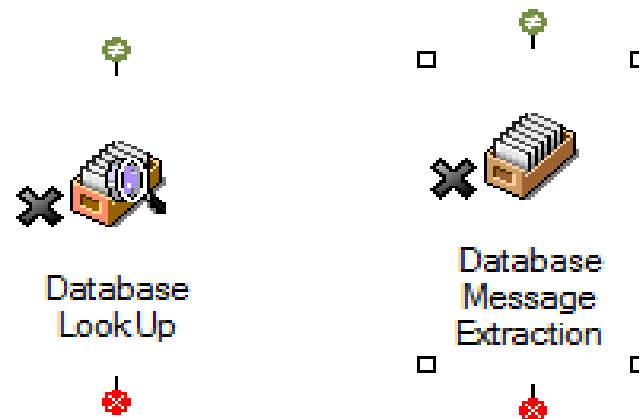
# Update Where Clause

- Identifies the condition that must be met before existing information in the database table will be updated with data from an incoming message
- In the example below an existing row will only be updated if the value in MSH.SendingFacility.NamespaceID field of an incoming message does not match the value in the IssuingAuthority column in the database table

	Column Name	Operator	Value	...	Type
1	IssuingAuthority	Not Equal	MSH.SendingFacility.NamespaceID	...	Field

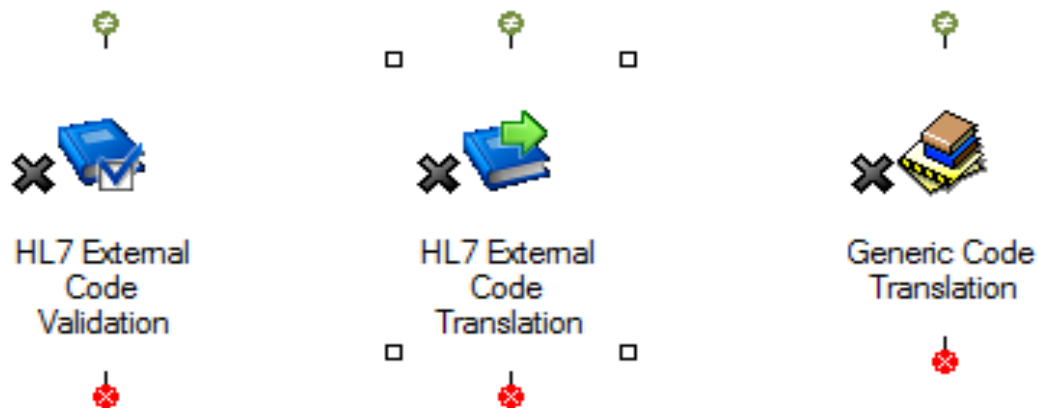
# Database Filters

- Database Lookup
  - Used to look up values in a record from a table and to create or update property values and field values as required
- Database Message Extraction
  - Replaces the body of the message with an XML message generated from database queries



# Database Filters Continued

- **Generic Code Translation**
  - Performs a database lookup to replace or populate the content of specified fields in a message
- **HL7 external code translation**
  - Similar to the Code Translation filter, optimized for translating codes in HL7 messages.
- **HL7 external code validation**
  - Similar to the HL7 External Code Translation filter, but is used to validate values and errors if they are invalid.



# Database Filter performance

- Database filters use the database to look up coded fields and can therefore become a bottleneck
- There are two strategies to improve performance
  - Connection Pooling
    - Allows the filter to hold multiple connections at the same time
    - If the pool size is too large other connections may not be able to connect to database
    - Recommended size is 2 to 5
  - Caching
    - Stores the coding systems that have been previously looked up which makes the subsequent lookups faster

# Database Configuration Editor

- External, Windows based component
- Provides a graphical interface for managing the database component configurations.
- Provides support for the databases which Rhapsody supports (Oracle, MS SQL Server and postgresSQL)

Mode:

Properties preceded by an asterisk (\*) are required.

Property	Value	...
Database	Manual Setting	▼
* Host		
Port		
* Database Name		
Username		
Password		
* Configuration File	Edit Configuration (Same as server)	...
Query Timeout	0	
Message Type		▼



Rhapsody Database Configuration Editor - Patient Details DB Out.xml

File Samples Help

Database Polling Options

## Patient Details DB Out

(in folder 0 - Database Output)

Database Communication Point (Output)

Patient DB Out SQLit...

**Record Exists ?**  
Check for existence of record

**Insert if doesn't exist (executes if `Equals(@record exists, 0)`)**  
`INSERT INTO PATIENTDETAILS (patientID, full_name, given_name, family_name, status, last_modified, modified_by) VALUES ( @PID.ExternalPatientID.IDNumber , $name , @PID.PatientName [0].GivenName , @PID.PatientName[0].FamilyName , 'Active' , datetime('now') , 'Rhapsody' )`

**Update Status (executes if `NotEqual(@record exists, 0)`)**  
Update the status if the record exists

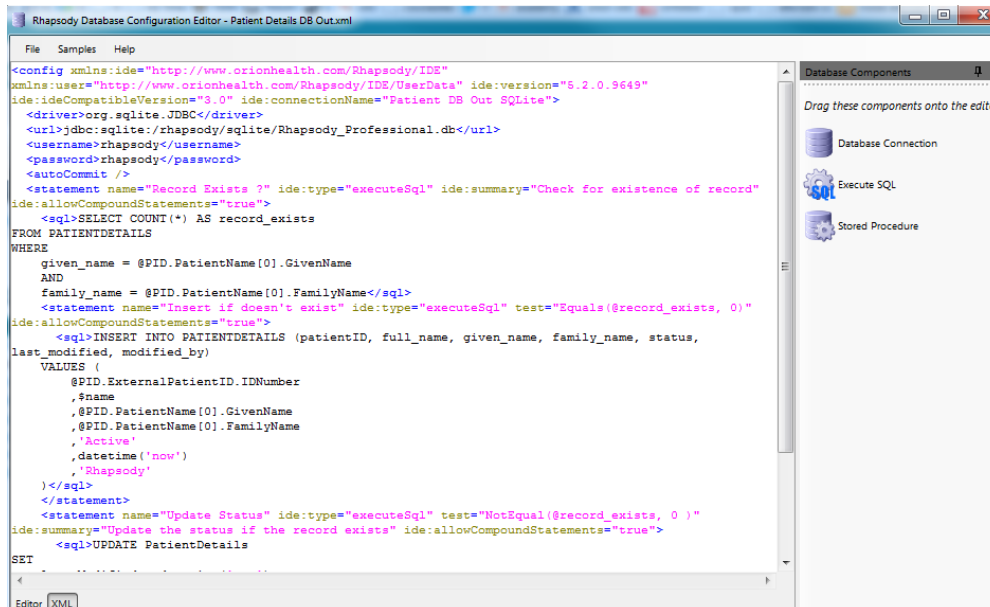
Database Components

Drag these components onto the editor

- Database Connection
- Execute SQL
- Stored Procedure

# Configuration Editor Interface

- **Follows the model for the Rhapsody IDE environment**
  - Menu bar
  - Used in database communication point and database filters
  - Toolbar containing common configuration requirements
  - The design canvas within which the configuration is displayed
  - Component menu on the right hand side
  - Ability to toggle between editor view and XML view



```
<config xmlns:ide="http://www.orionhealth.com/Rhapsody/IDE"
xmlns:user="http://www.orionhealth.com/Rhapsody/IDE/UserData" ide:version="5.2.0.9649"
ide:ideCompatibleVersion="3.0" ide:connectionName="Patient DB Out SQLite">
  <driver>org.sqlite.JDBC</driver>
  <url>jdbc:sqlite://rhapsody/sqlite/Rhapsody_Professional.db</url>
  <username>rhapsody</username>
  <password>rhapsody</password>
  <autoCommit />
  <statement name="Record Exists ?" ide:type="executeSql" ide:summary="Check for existence of record"
ide:allowCompoundStatements="true">
    <sql>SELECT COUNT(*) AS record_exists
FROM PATIENTDETAILS
WHERE
    given_name = @PID.PatientName[0].GivenName
    AND
    family_name = @PID.PatientName[0].FamilyName</sql>
    <statement name="Insert if doesn't exist" ide:type="executeSql" test="Equals(@record_exists, 0)"
ide:allowCompoundStatements="true">
    <sql>INSERT INTO PATIENTDETAILS (patientID, full_name, given_name, family_name, status,
last_modified, modified_by)
VALUES (
    @PID.ExternalPatientID.IDNumber
    ,%name
    ,@PID.PatientName[0].GivenName
    ,@PID.PatientName[0].FamilyName
    ,'Active'
    ,datetime('now')
    ,'Rhapsody'
) </sql>
  </statement>
  <statement name="Update Status" ide:type="executeSql" test="NotEqual(@record_exists, 0 )"
ide:summary="Update the status if the record exists" ide:allowCompoundStatements="true">
    <sql>UPDATE PatientDetails
SET
```

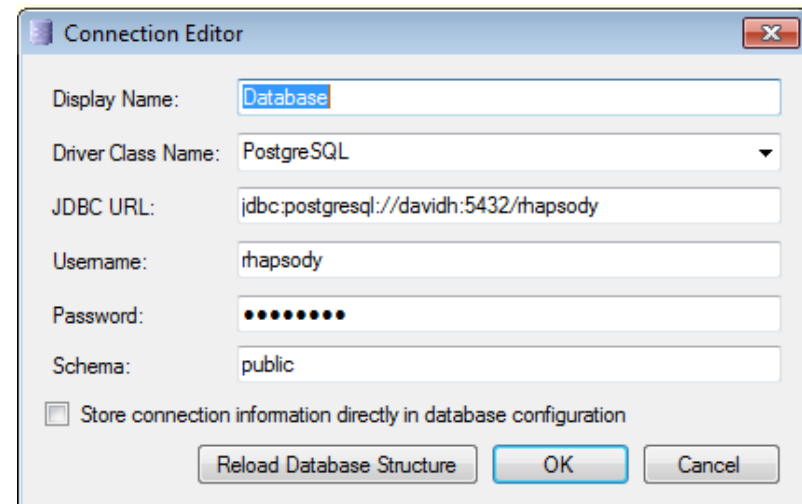


# Creating a New Configuration

- The process for creating a configuration is as follows
  1. Define the details Database Connection
  2. Define the Polling parameters (input modes only)
  3. Define the Options
  4. Build the required Statements
  5. Save and Exit from the Editor
  6. Check in the component and test.

# Defining the Connection Details

- Driver Class: the driver specific name required for the driver
- JDBC connection string: the location of the database that Rhapsody will connect to (defined in a database specific format)
- Username
- Password



The screenshot shows a 'Connection Editor' dialog box with the following fields and options:

- Display Name: Database
- Driver Class Name: PostgreSQL
- JDBC URL: jdbc:postgresql://davidh:5432/rhapsody
- Username: rhapsody
- Password: [masked with dots]
- Schema: public
- Store connection information directly in database configuration
- Buttons: Reload Database Structure, OK, Cancel

# Support for Database Drivers

- Rhapsody environment provides built-in support for a number of databases (MS SQL Server, Oracle and postgresSQL)
- Support for other databases requires that a JDBC driver be available
- Utilizing another JDBC driver requires that both the engine and the Database Configuration Editor are able to access the driver

# Connection Details

- **Good Practice**
  - Use Rhapsody Variables for consistency
    - JDBC URL
    - Username
    - Password
  - Protects details as the configuration is migrated between hosts
  - Specify in Configuration Tab if using Rhapsody Variables

# SQL Statement Editor

- Work of database components carried out by statements
- Each statement is named with the name utilized in the structure of the output
- Each statement contains one or more SQL statements to be executed against the database



## Record Exists ?

Check for existence of record



## Insert if doesn't exist (executes if `Equals(@record_exists, 0)`)

```
INSERT INTO PATIENTDETAILS (patientID, full_name, given_name, family_name, status, last_modified, modified_by) VALUES ( @PID.ExternalPatientID.IDNumber , $name , @PID.PatientName [0].GivenName , @PID.PatientName[0].FamilyName , 'Active' , datetime('now') , 'Rhapsody' )
```



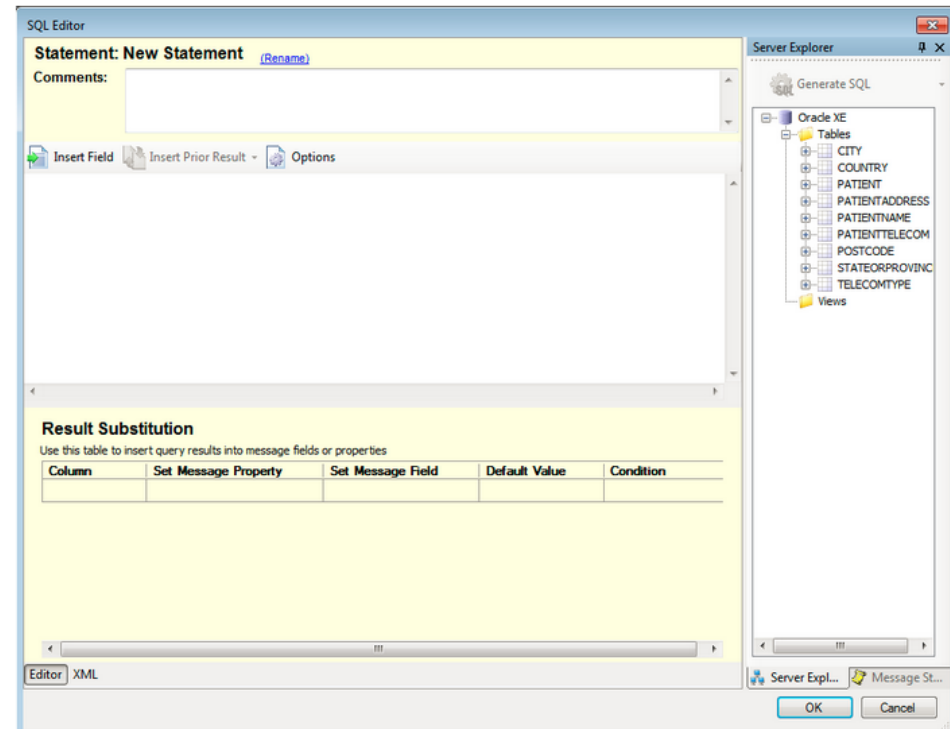
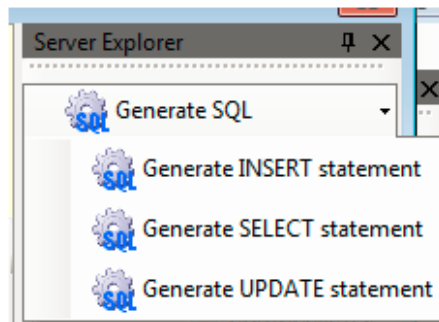
## Update Status (executes if `NotEqual(@record_exists, 0)`)

Update the status if the record exists

# SQL Statement Editor Continued

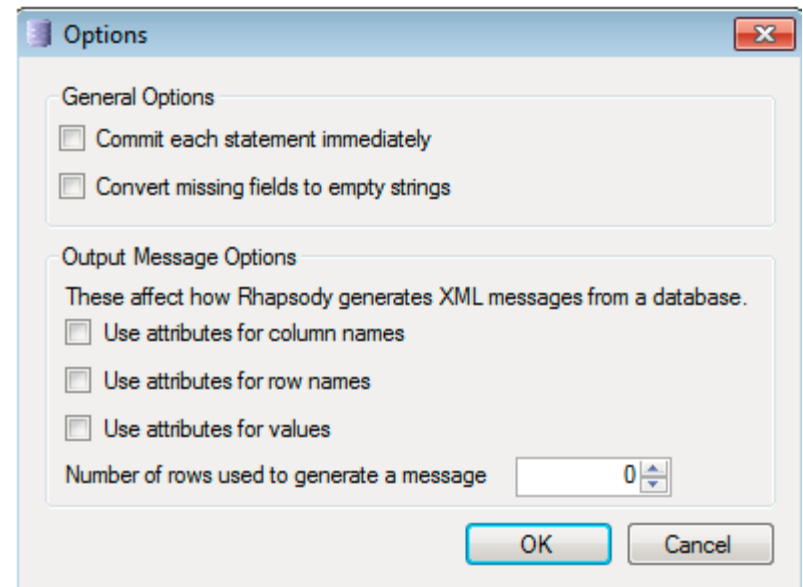
- **Key Features**

- Browse the database schema
- Auto generate statements from schema (select, insert, update)
- Insertion of fields from a message definition via drag and drop
- Update fields in the message automatically
- Set message properties from fields returned by the statements
- Define a stored procedure output parameter
- Reference the complete message body



# Database Options

- The default execution mode for the database components is to execute all of the configured statements as a transaction
- Convert missing fields to empty strings ensures that all fields will be returned by the statements with at least an empty string
- The Output Message options provide control of the format of the XML message returned by the statements










# Stored Procedure DEMO



Thank you for joining us today, for additional questions regarding Rhapsody....

You can contact us through our forum at

<http://interfaces.galenhealthcare.com/>

RHAPSODY INTERFACES		TOPICS	POSTS	LAST POST
 Reg/Sched/Charge	0	0	No posts	
 Order/Result	0	0	No posts	
 Document/Transcription/Dictation	0	0	No posts	
 Immunization/Medication	0	0	No posts	
 ConnectR to CIE Migrations	0	0	No posts	
 Errors/Troubleshooting	0	0	No posts	
 Tools/Scripts/Filters/Best Practices	0	0	No posts	