LYNIATE

Optimizing and Managing Your Environment

Connect 22

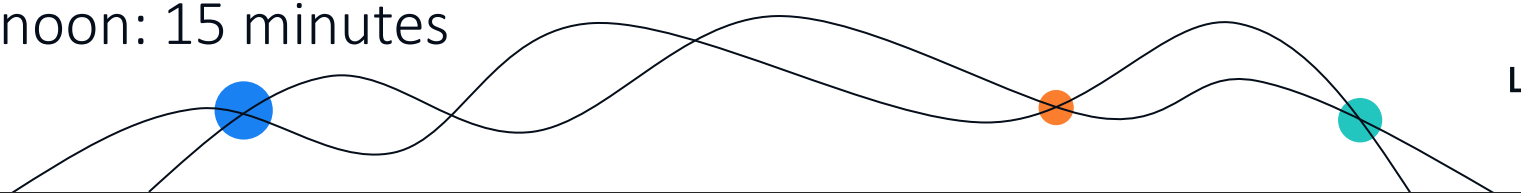
What to expect….

Full Day:

- 11:00 a.m. – 6:00 p.m. NZDT
- 9:00 a.m. – 4:00 p.m. AEDT
- 8:00 a.m. – 3:00 p.m. AEST
- 6:00 a.m. – 1:00 p.m. SGT

Breaks:

- Morning: 15 minutes
- Lunch: 30 minutes
- Afternoon: 15 minutes



Class Setup

Instructor

Rhapsody Optimization and Using APIs

Classroom
Instructor Presentation & Group Discussions

Rhapsody Integration Engine
Hands-on Activities

Training Attendees

LYNIATE

Class Interactions

Don’t be shy!

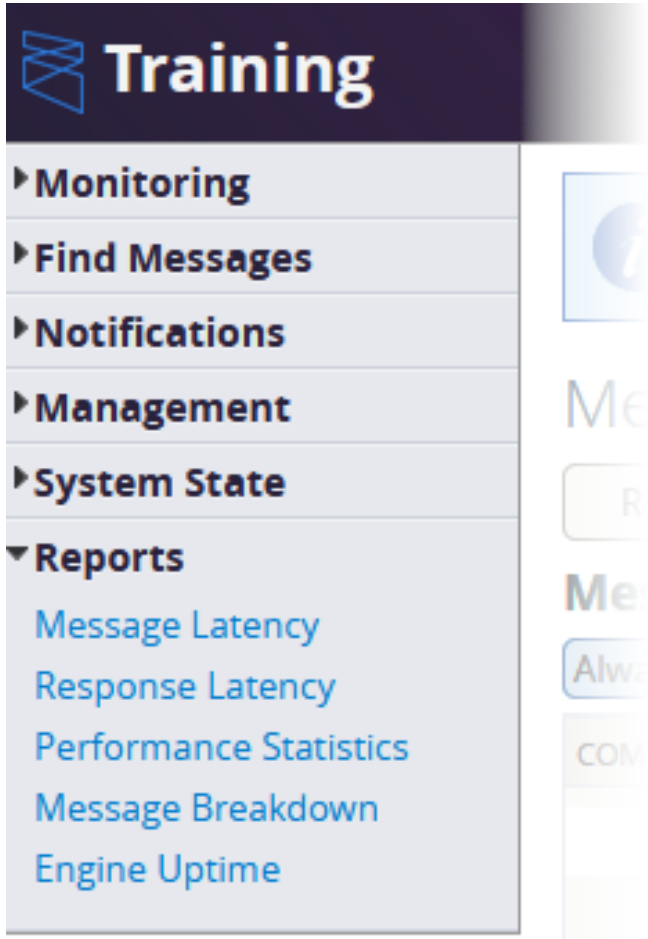
- Please ask questions
- Answer questions the instructor asks, so he doesn’t get lonely up there
- Comment on something the instructor says
- Have fun while learning

What are you going to learn?

- System Reports
- Best Practices
- Defensive Configuration

System Reports

Management Console > Reports



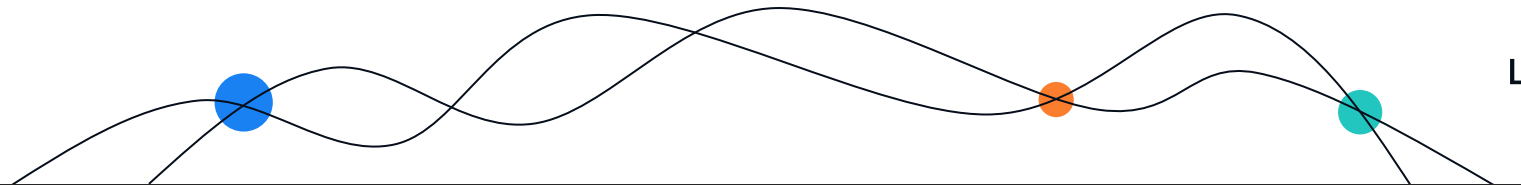
LYNIATE

Exercise – System Reports

LYNIATE

Exercise

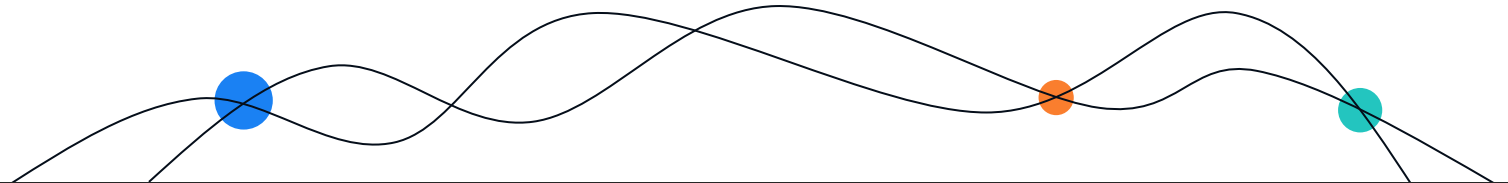
- Run the “Message Latency” report for the “ArchiveAllPASMessages” output
  - What is the mean time for message sent to this communication point?
- Open the “Response Latency” report
  - What two communication points are we able to select to run this report against?
- Start the Performance Report running
  - Name the top 4 routes in terms of time
  - Looking a the “Getmessages” route, what three filter account for most of the processing time?
- Run the Message Breakdown report
  - What type of message is being processed by the configuration?
  - What is the name of the communication point where these files are being received?



Best Practices

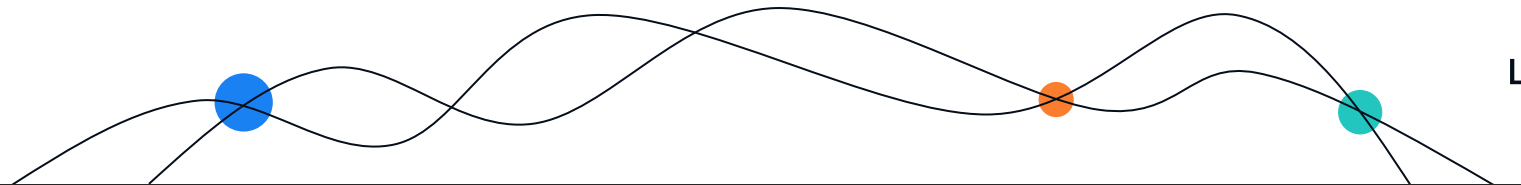
Benefits

Best Practice	Benefit
Consistent Delivery Patterns	Consistency across solutions and projects so Development and Support teams can readily understand the message flow.
Reduced Effort	Rhapsody offers multiple tools that may be used to solve the same types of problems. By formalizing the known optimal patterns, time and effort testing between possible solutions is reduced.
Optimal Outcomes	Leveraging lessons of experience ensures optimal solutions and avoids known bad patterns that often result in support issues or sub-optimal processing and performance.



Objectives

- ✓ Understandable
- ✓ Maintainable
- ✓ Performant

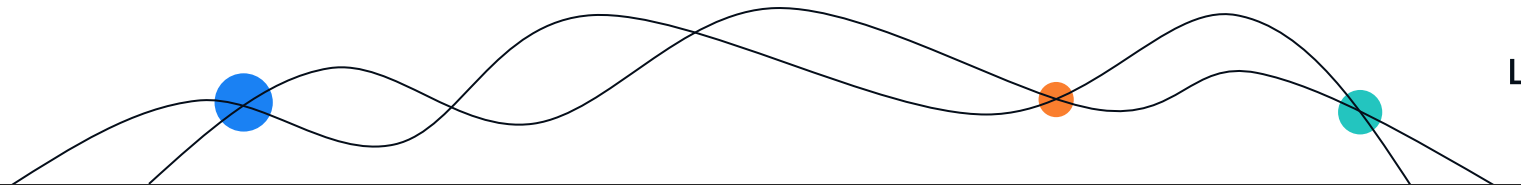


Understandable

What do we mean by Understandable?

- ✓ Clearly laid out in a visual sense
- ✓ The purpose is stated through naming, notes or other easily discerned ways
- ✓ The complexity level of any single component is minimized

Ask yourself: *Would I understand this configuration 12 months after last reviewing or working on it?*

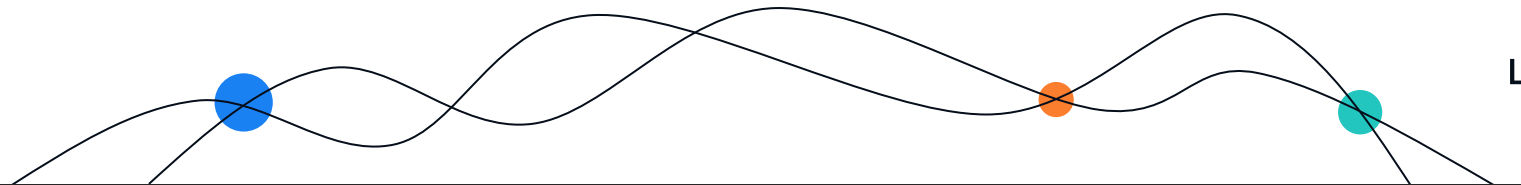


Maintainability

To achieve maintainability the configuration should be:

- ✓ Understandable
- ✓ Self-documenting
- ✓ Easily packaged for upstream deployment
- ✓ Testable

Ask yourself: *How long would it take for another developer to break-fix this project?*

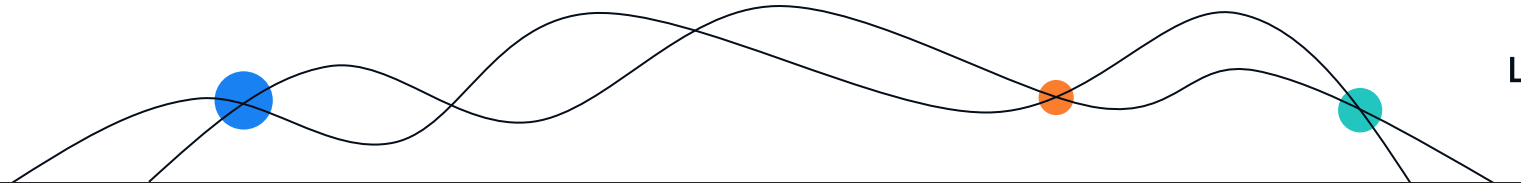


Performant

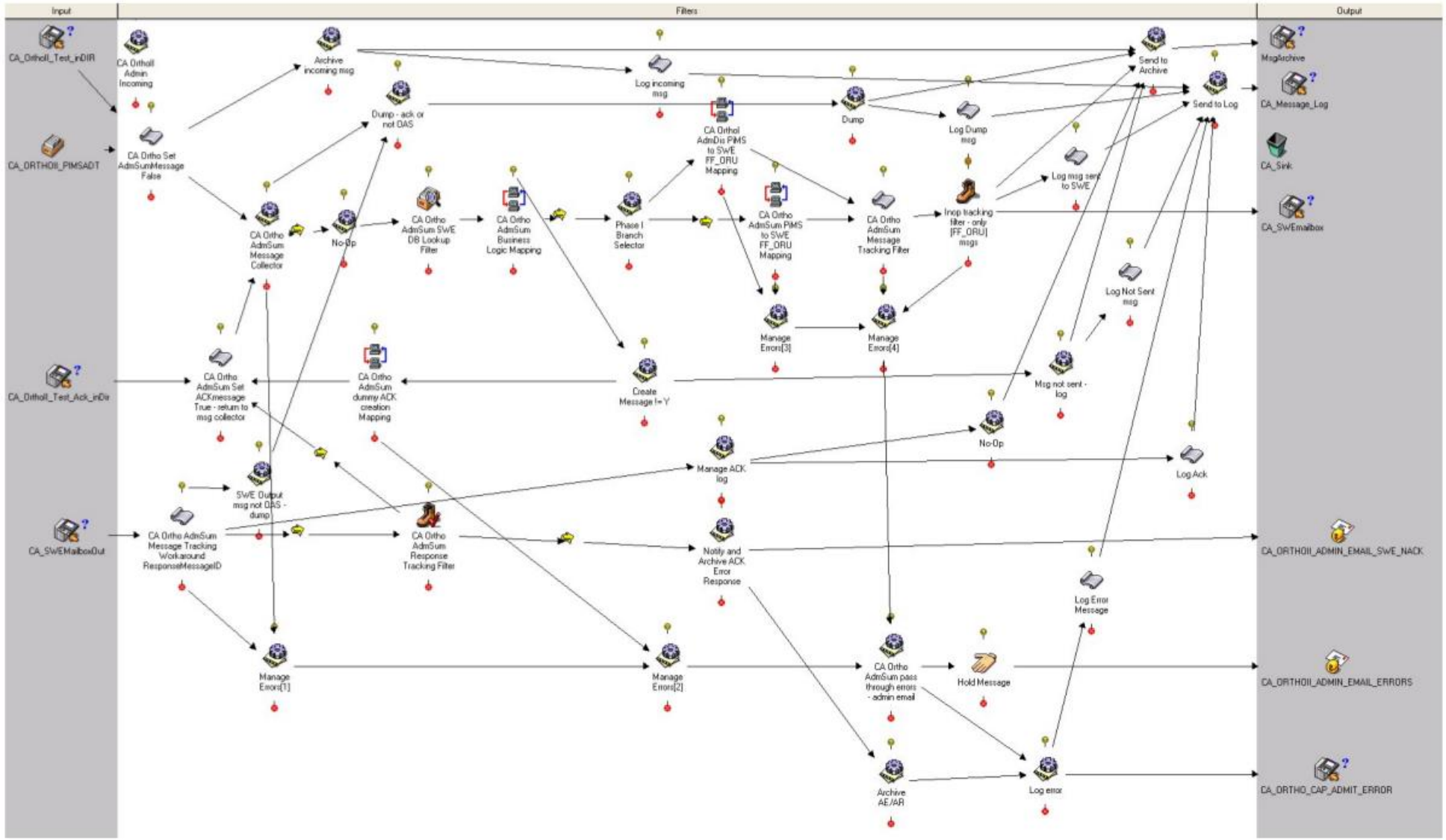
How performant is your project? You can examine from several vectors:

- ✓ Route design, single vs multiple, re-use or replicate?
- ✓ Can FIFO be disabled? Is multi-threading a good option?
- ✓ What alternative components can be used to solve the same problem?
- ✓ How does this project impact engine resources? How can I find out?

Ask yourself: *Do I know what to expect when we Go-Live or will I be surprised?*

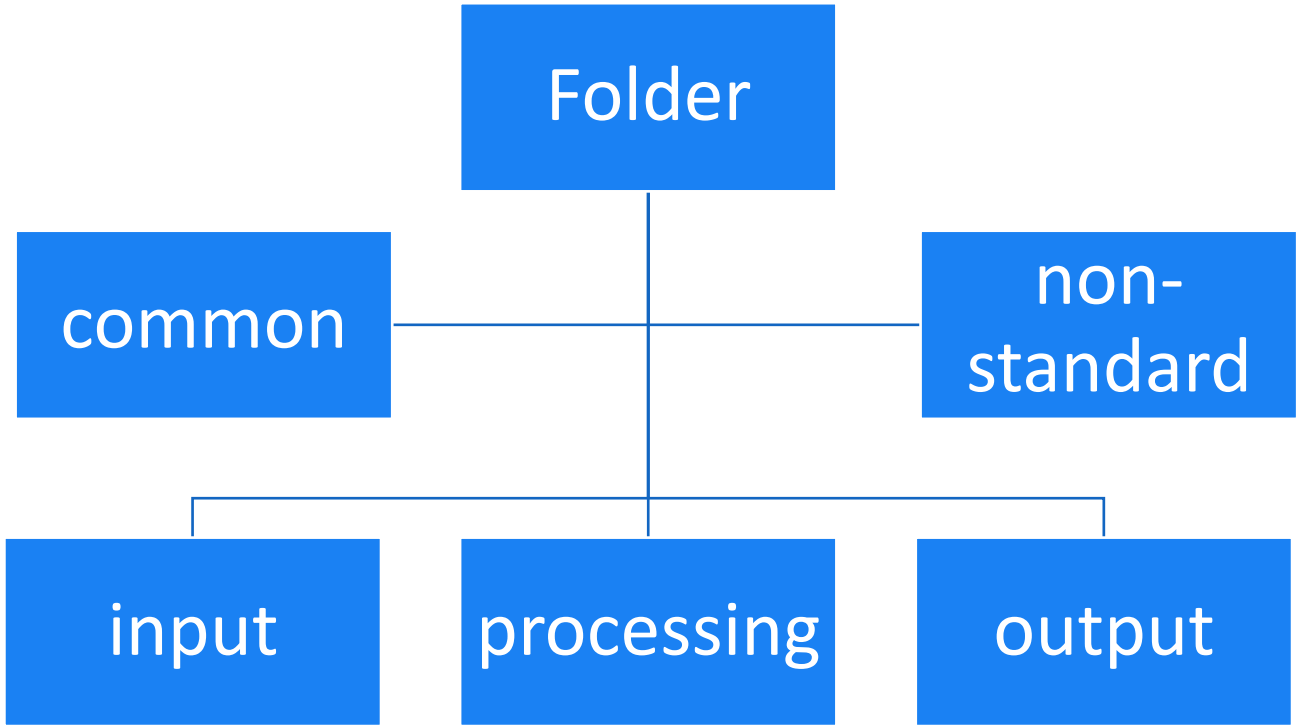


Route Layout



LYNIATE

SOA/Decoupled Paradigm



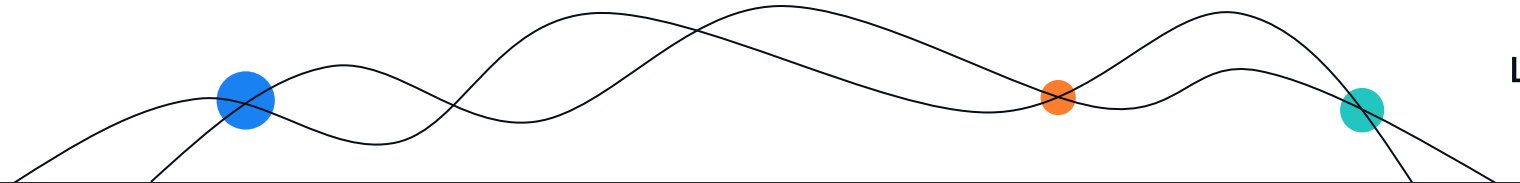
Lockers

- To segregate components to ensure team members are not able to modify components that are not related to their solution or tenancy.

- To ensure that administrative users are only able to access a subset of message data and components within the management console (e.g., to restrict access to patient data only from their site, or to ensure they are only able to start/stop components under their jurisdiction).

Route Design

- Before working on a new Rhapsody solution, it is worth spending time to develop a high- level overview of its intended purpose and to establish a naming convention for its lockers, folders, routes and communication points. This will help identify which components belong together and will assist when searching for a specific item.
- If there are several filters on the route, each of which changes the message, a copy of the message will be saved each time the message exits a filter. This can increase the processing load on the engine. As a result, we recommend that routes are designed with the minimum number of filters needed to fulfil their requirements.
- When building a route, keep in mind that it should be as simple as possible yet still meet all site requirements



Route Layout

- Keep the message flow left to right
- Layout is logical and easy to interpret 'at-a-glance’
- Connectors should not cross each other and should generally not be aligned vertically
- There should generally only be one definition on a route

Naming Conventions 1

- Purpose
  - Label components in a manner which is clear and indicative of purpose.
  - Create configurations which are self-documenting
  - Make configuration easy to find
- If a name occupies more than three lines in the Route Canvas, shorten it
- Spaces are preferred over underscores in a name

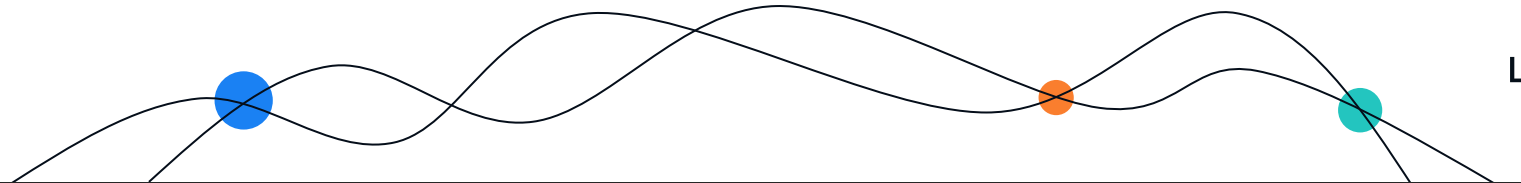
Naming Conventions 2

Lockers and Folders

- Identifies the solution, project (or part of project) they contain

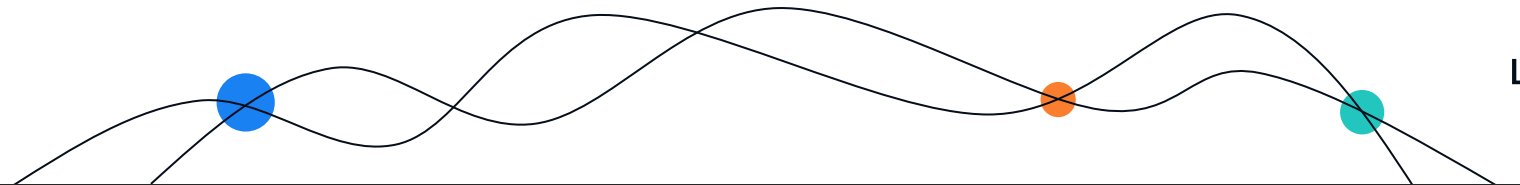
Communication Points

- Named to help identify their function and purpose
- Consider:
  - Identifying the direction of message movement; for example *Input* means that Rhapsody is receiving the message while *Output* means that Rhapsody is sending the message.
  - Searching for a component in the **Management Console**
  - Analysing and understanding message processing logic



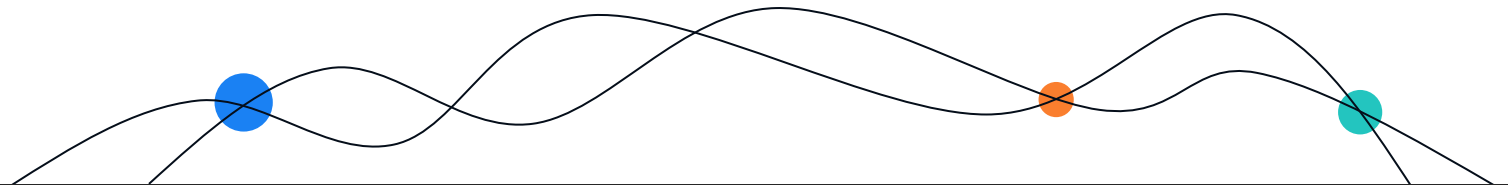
Naming Conventions 3

- Routes
  - Indicate their high level purpose
- Filters
  - Action being performed on message
- Message Definitions
  - Name should closely reflect the message standard and version they define
- Map Definitions
  - Identify purpose. Normally in the form *source-destination-messages*.



Documentation

- All routes should contain notes that include author, date, and purpose and history details
  - The documentation should detail the intended purpose of the route, along with any non-obvious aspects of the configuration
  - Include links to specifications or additional material
  - Think about formatting for example, **bold** & *italic* text
- Documentation for routes, communication points, & filters is included as part of configuration PDF
- Use a standard template to make formatting and inclusion consistent
- Support Notes are created for communication points that connect to an external system
  - Provides relevant information to support to diagnose a connection problem.



JavaScript & Mapper code comments

- Purpose Comment
- Change History
- Submap Comments
- Use inline comments for custom or complex code

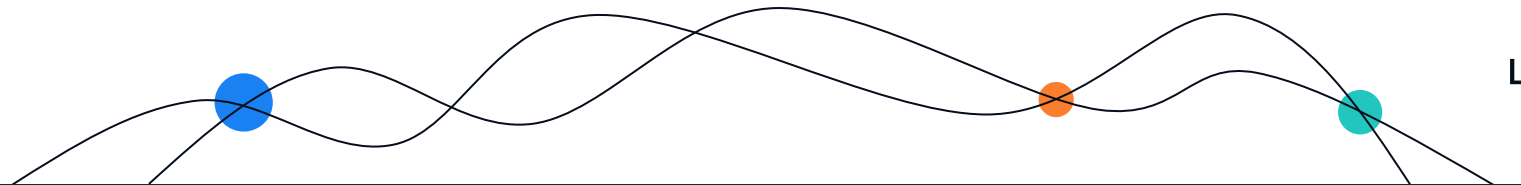
```
map mainADTA04( <- Input::ADTA04 in, -> Output::ADTA04 out )
{
    /*
        Map ADT^A04 message received from General Hospital Systems to the
        corrisponding A04 message required by SimplePAS. SimplePAS only wants the
        following segments in the received messages:
         - MSH
         - EVN
         - PID
    */

    MapMSHToMSH(in.MSH, out.MSH);
    MapEVNToEVN(in.EVN, out.EVN);
    MapPIDToPID(in.PID, out.PID);
}
```

```
map ChangeHistory( -> string out )
{
    /*  Change History

        2018-02-26 - Thomas Slacker - Add Main map for A04, as vendor will now be sending them.
        2018-10-19 - Rowena Tazik - Remove all SSN from output per receiving vendor request.
        2019-03-26 - Lie Nguyen - Change OID for MSH-5.2
    */
}
```

```
map MapPIDToPID( <- Input::PID in, -> Output::PID out )
{
    /*
        We only need to map a minimum set of fields in the PID segment as SimplePAS
        only uses identifiers (including SSN), name, sex, & dob to match an existing
        patient.
    */
```

Testing

- ✓ Test all complex filter configuration
- ✓ Test both success and failure scenarios
- ✓ Create test functions for Shared JavaScript functions
- ✓ Use Test Manager to check coverage of configuration



Exercise – Configuration Review

LYNIATE

Exercise

HL7 ADT messages received from Patient Administration System

Messages sent to the following four systems:

- Ward Info (via TCP)
- PACS (via TCP)
- Simple Lab (via REST Web Service)
- Mental Health (via SOAP Web Service)

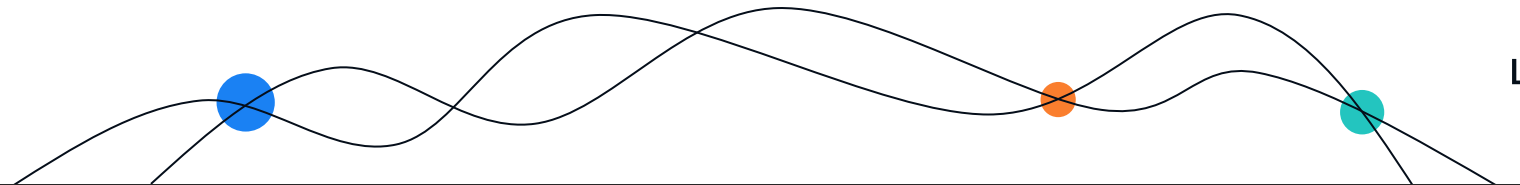
Use Papercut to view email messages. More messages can be sent with EDI Explorer.

Tasks:

- Note down the good and bad practices you find
- Note anything in the configuration you would like discussed

Exercise – The Good

- Use of folders to group related components
- Test configuration (in “Test Systems” Locker)
- Layout and naming of “Simple Lab” route
- Filter testing configuration in “Simple Lab” route
- Test function on shared JavaScript function
- Use of Dynamic Router to link to error messages
- Message tracking for messages sent to Wand Info system
- Documentation on “Simple Lab”, “Ward Info” and “Error Processing” routes



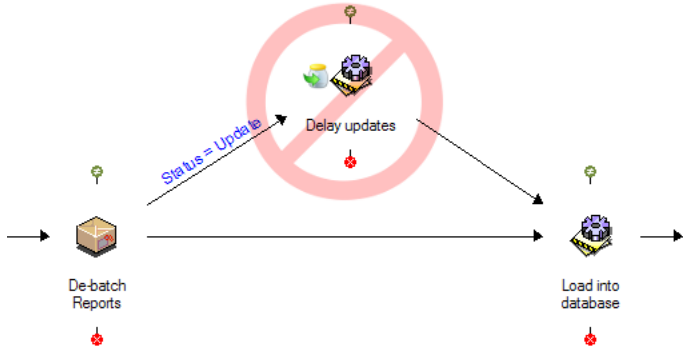
Exercise – The Bad

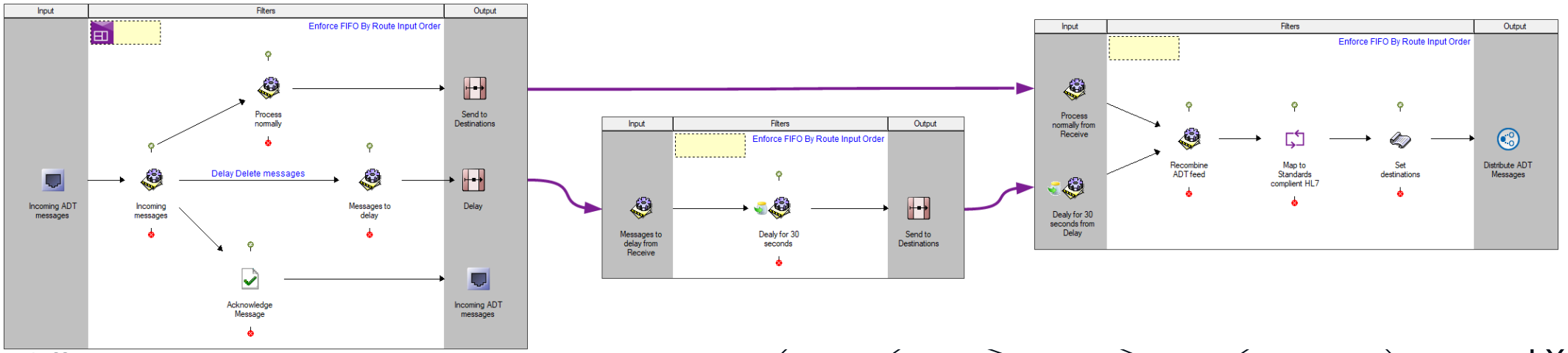
- Use of default component names
- Lack of documentation
- Messages sent to error queue from “MH” route
- TCP communication points used to connect routes (between “Getmessages” and “NewRoute”)
- Layout of “NewRoute” route
- Timed hold message collector on “NewRoute” won’t have any impact on message order
- Chained JavaScript filters on “Getmessages” route
- PAS TCP server communication point used in two places (“Getmessages” and “Acknowledgement”)
- Components with a start state of manual (“MH” route)

Best Practices II

Truisms

- Use input Communication Points only once
- Where possible de-batch on a directory communication point
- Don’t store the message body as a property
  - (Snapshots are a better alterative)
- Consider templates for frequently used configuration
- Don’t delay messages on a route that uses FIFO to preserve order





LYNIATE

Connecting routes

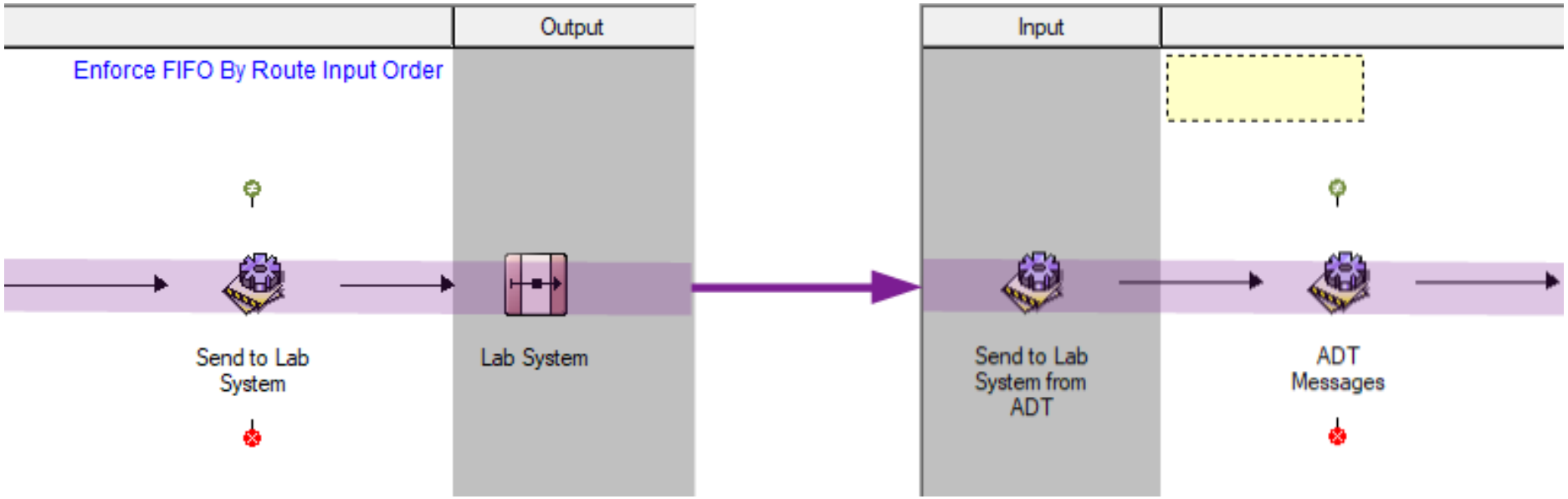
Methods

- Direct Connection
- Dynamic Router communication point
- Rhapsody Connector communication point

Tightly coupled

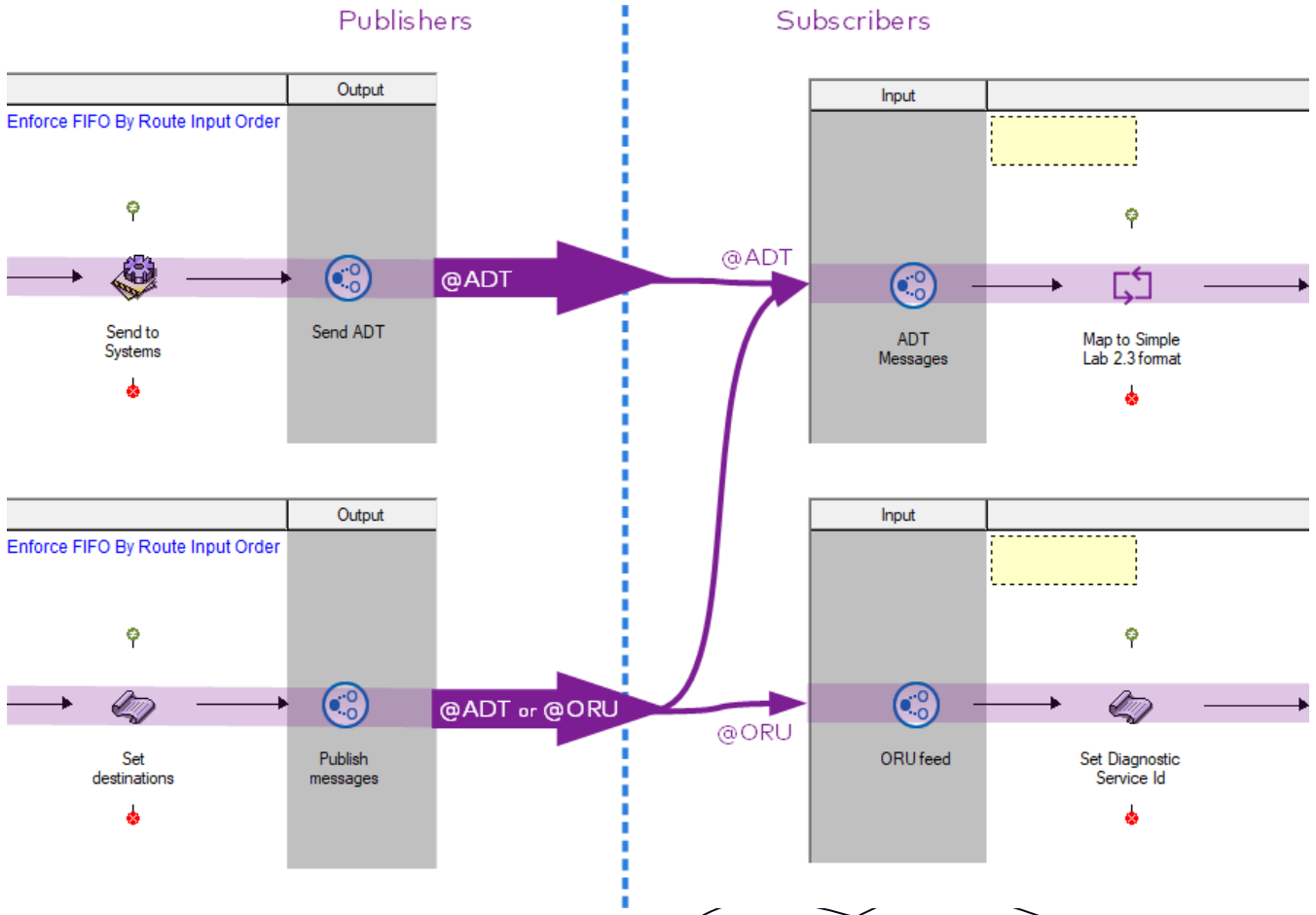
Loosely coupled

Direct Route Connection

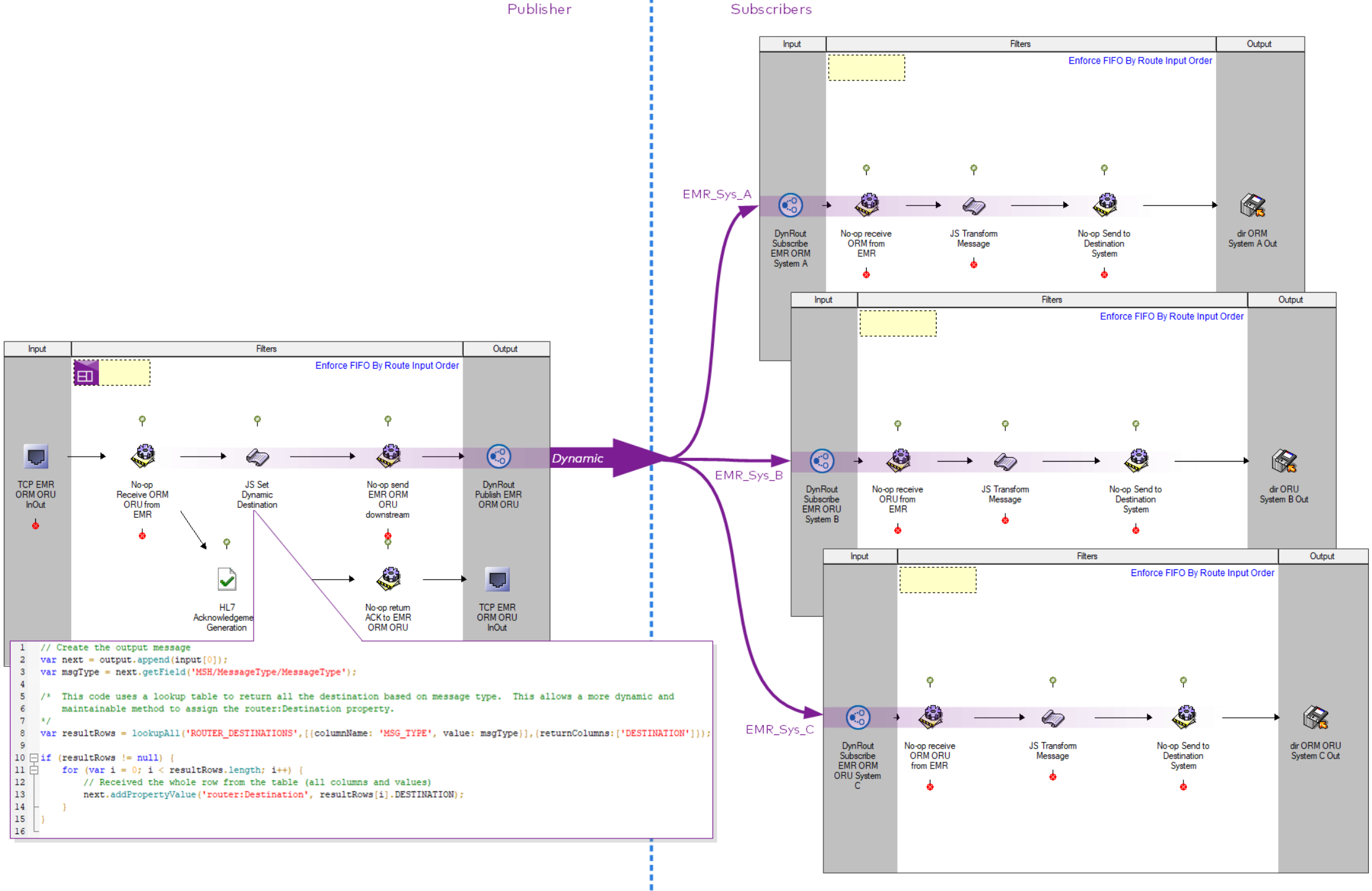


LYNIATE

Dynamic Router Communication Point



LYNIATE


Publisher
Subscribers
Input
Filters
Output
Enforce FIFO By Route Input Order
TCP EMR ORM ORU InOut
No-op Receive ORM ORU from EMR
JS Set Dynamic Destination
No-op send EMR ORM ORU downstream
DynRout Publish EMR ORM ORU
HL7 Acknowledgeme Generation
No-op return ACK to EMR ORM ORU
TCP EMR ORM ORU InOut
Dynamic
EMR_Sys_A
EMR_Sys_B
EMR_Sys_C
DynRout Subscribe EMR ORM System A
No-op receive ORM from EMR
JS Transform Message
No-op Send to Destination System
dir ORM System A Out
DynRout Subscribe EMR ORU System B
No-op receive ORU from EMR
JS Transform Message
No-op Send to Destination System
dir ORU System B Out
DynRout Subscribe EMR ORM ORU System C
No-op receive ORM ORU from EMR
JS Transform Message
No-op Send to Destination System
dir ORM ORU System C Out
// Create the output message
var next = output.append(input[0]);
var msgType = next.getField('MSH/MessageType/MessageType');
/* This code uses a lookup table to return all the destination based on message type. This allows a more dynamic and
maintainable method to assign the router:Destination property.
*/
var resultRows = lookupAll('ROUTER_DESTINATIONS',[{columnName: 'MSG_TYPE', value: msgType}],{returnColumns:['DESTINATION']});
if (resultRows != null) {
for (var i = 0; i < resultRows.length; i++) {
// Received the whole row from the table (all columns and values)
next.addPropertyValue('router:Destination', resultRows[i].DESTINATION);
}
}

Comparison - Advantages

Direct Route Connection

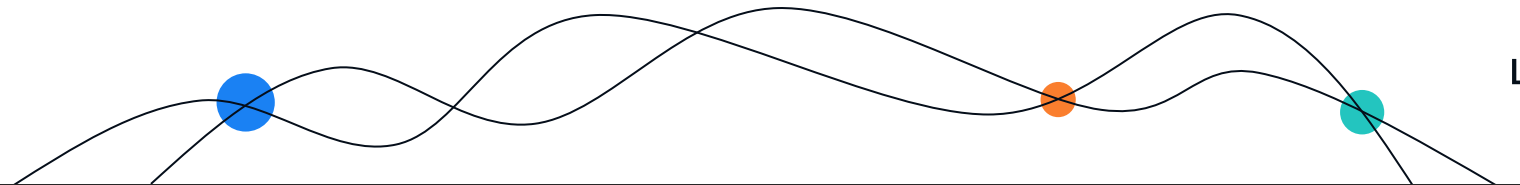
Advantages

- Simple to drag and drop destination route onto source route
- Double-clicking the route icon in the Input or Output section of the route links to the connected route.
- Message properties are preserved.
- Continual tree view in the Management Console.

Dynamic Router Communication Point

Advantages

- One Publisher Dynamic Router can send to multiple subscribing routes, making migrating the configuration less complicated.
- Message properties are preserved.
- Continual tree view in the Management Console.
- Messages can be sent across Lockers.



Comparison - Disadvantages

Direct Route Connection

Disadvantages

- When connecting multiple routes, the source route Output section can get overcrowded and difficult to maintain.
- When migrating configurations, having depended routes can make the migration more difficult.

Dynamic Router Communication Point

Disadvantages

- Naming convention needs to be clear, otherwise it may not be obvious what the destination for a message sent to a dynamic router will be.
- Without careful design, it is possible to create an infinite loop between routes.

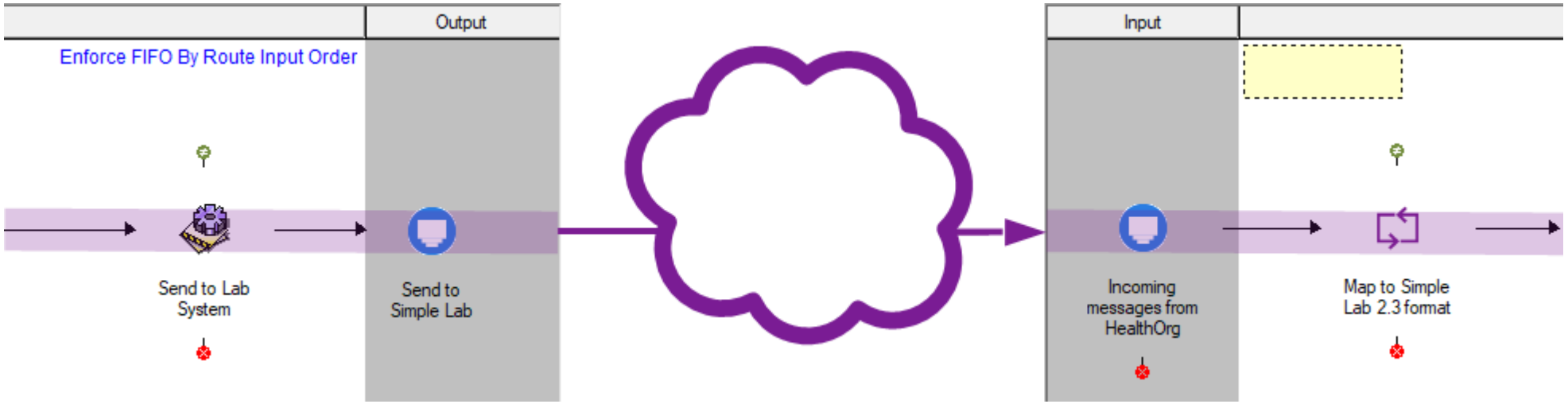
Multi-Threading

Rhapsody is a multi-threaded Java application and uses 10 worker threads (by default) to manage all the route processing.

In general, the processing cycle is:

- Route worker thread becomes available and is in the “waiting for message” state
- Thread pulls the next available message from an input queue
- Thread processes the message from input to output (white canvas.)
- Processing on the last route filter completes and the message is delivered to the output (Communication Point or Linked Route)
- Route execution finishes
- Route worker thread becomes available once again

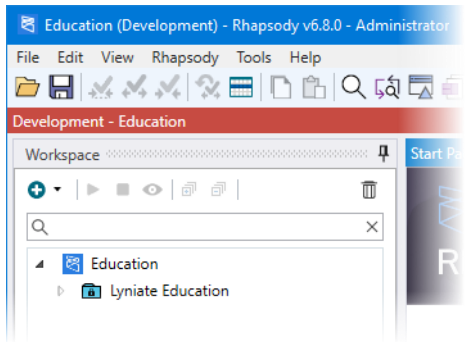
Connecting to other Rhapsody Engines

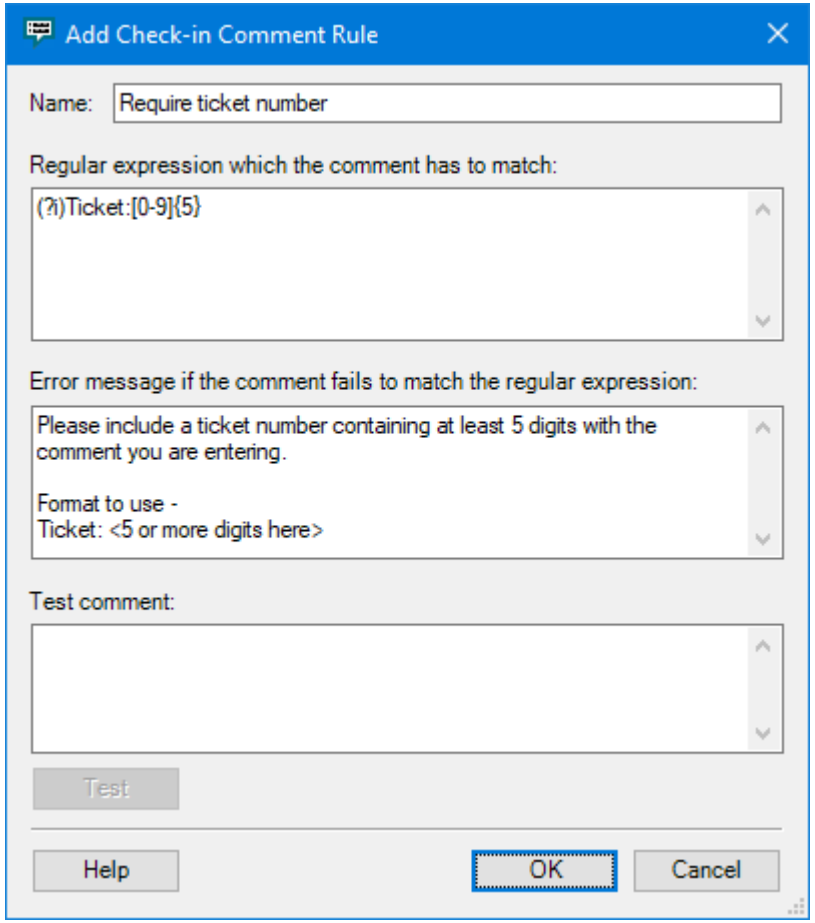


LYNIATE

Managing the Configuration

- Create accounts for each user
- Create check-in comment rules for non-development environments
- Monitor lookup table failures
- Use the Environment Indicator to highlight the name of each environment

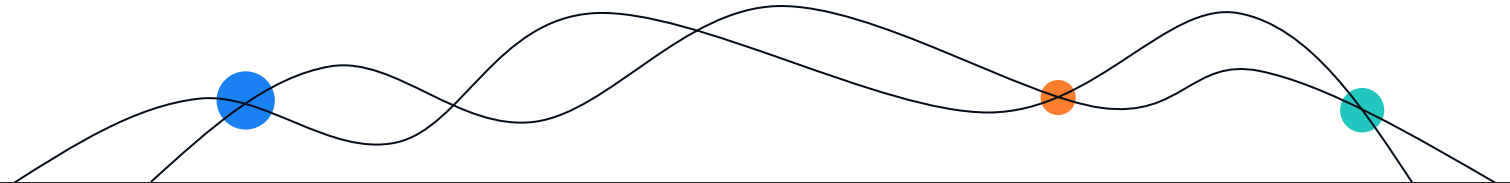




LYNIATE

Exceptions to Best Practices

- When adhering to one best practice principle is at odds with another – in such situations, adhere to the principle of greatest importance for that scenario.
- A novel solution may be required that requires a pattern that has not been widely encountered and has not been considered.
- Practical or licensing constraints, or directives from vendor or exchange exchange partner.
- Ensure that the approach is sound.
- Foster discussion and review from a larger pool of experienced stakeholders.
- Ensure future similar patterns that deviate from best practices will be handled in a consistent manner.
- Ensure the proposed interface design is in alignment with the intended use of Rhapsody and rules of your organization.



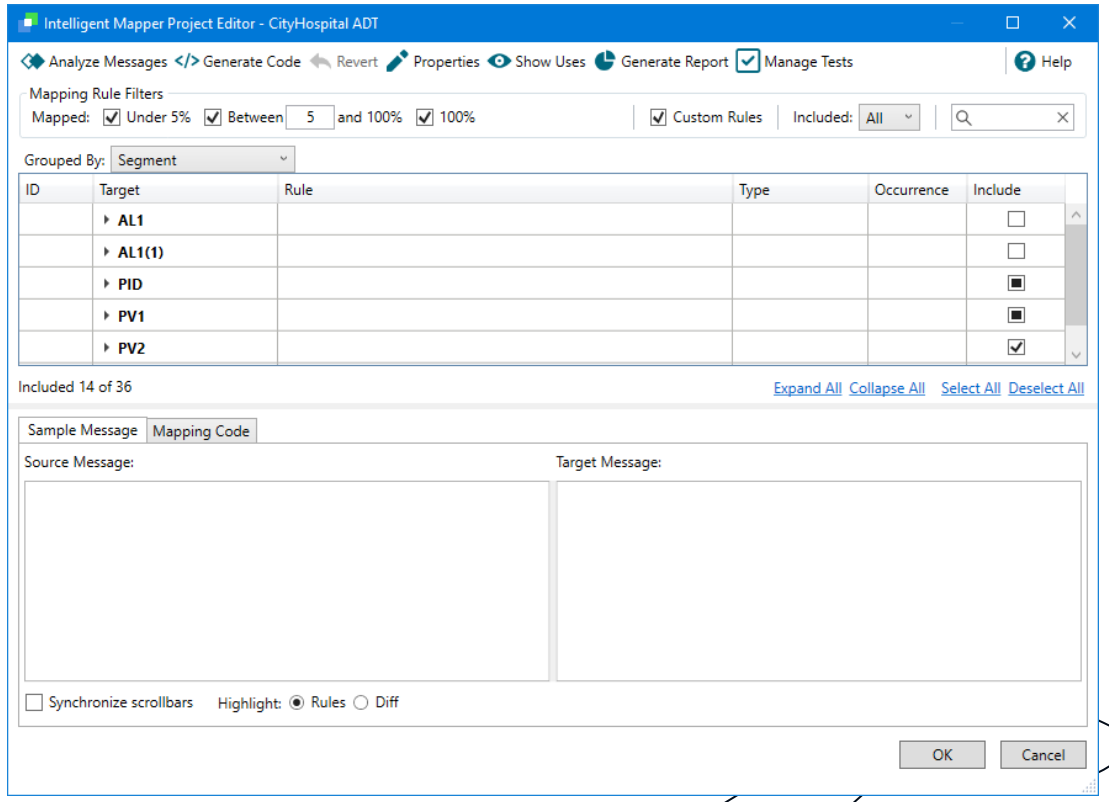
Defensive Configuration

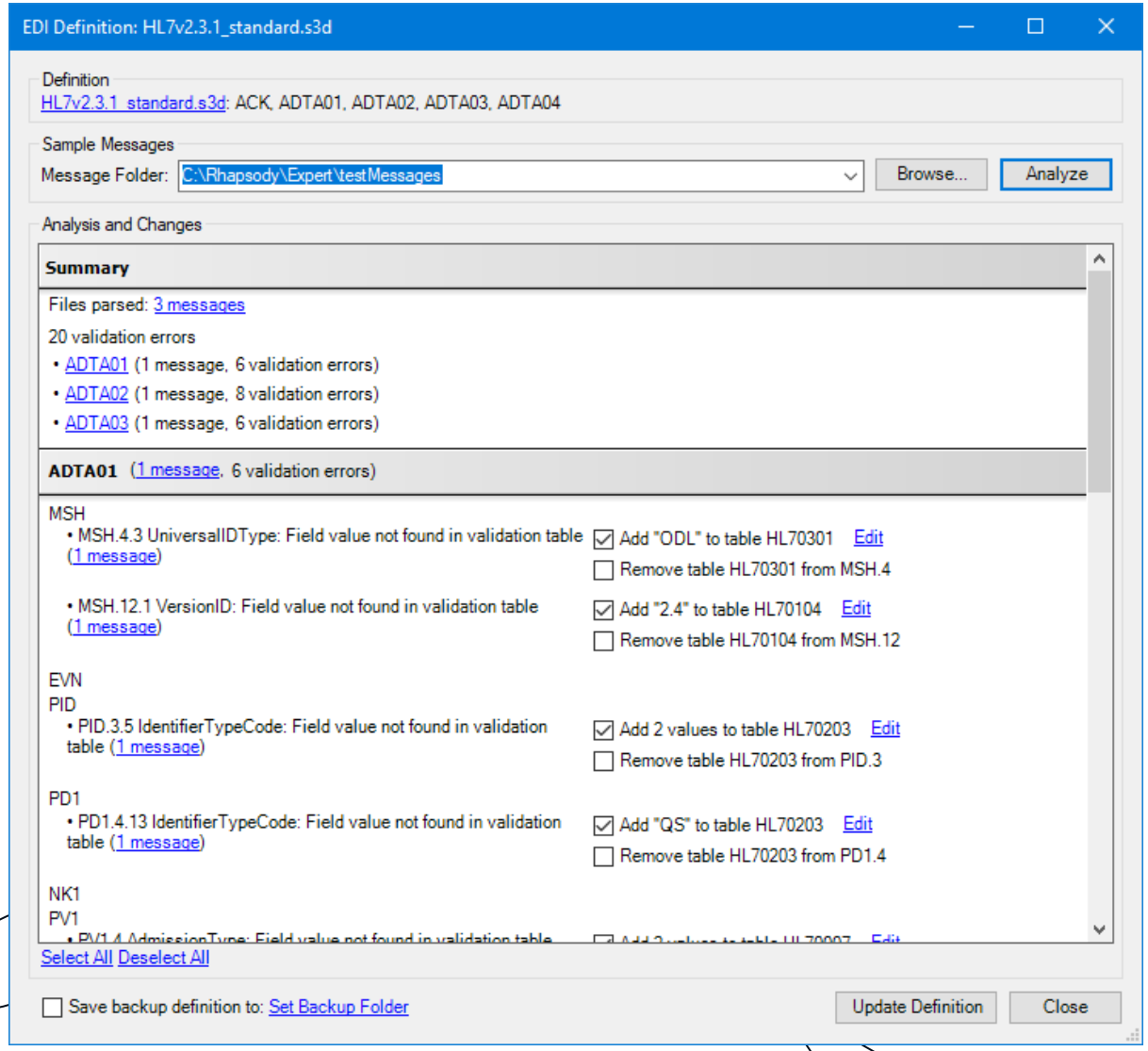
Principles

- ❑ Anticipate all error conditions
- ❑ Prevent messages being sent to error queue
- ❑ Supports maintainable and performant best practice goals

Know what Messages are being Received

- Adapt to Message
- Intelligent Mapper





Manage All Errors

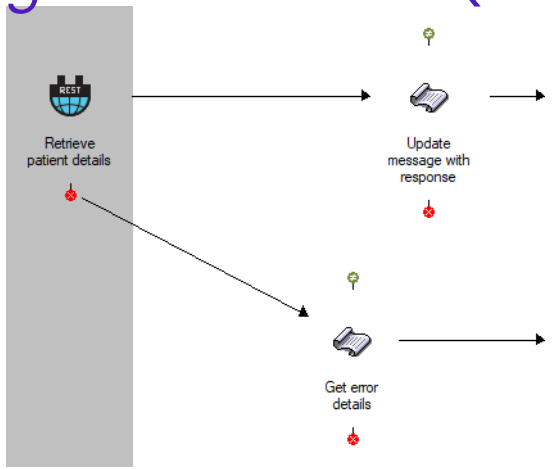
In production, the goal should be to have no messages on the Error Queue

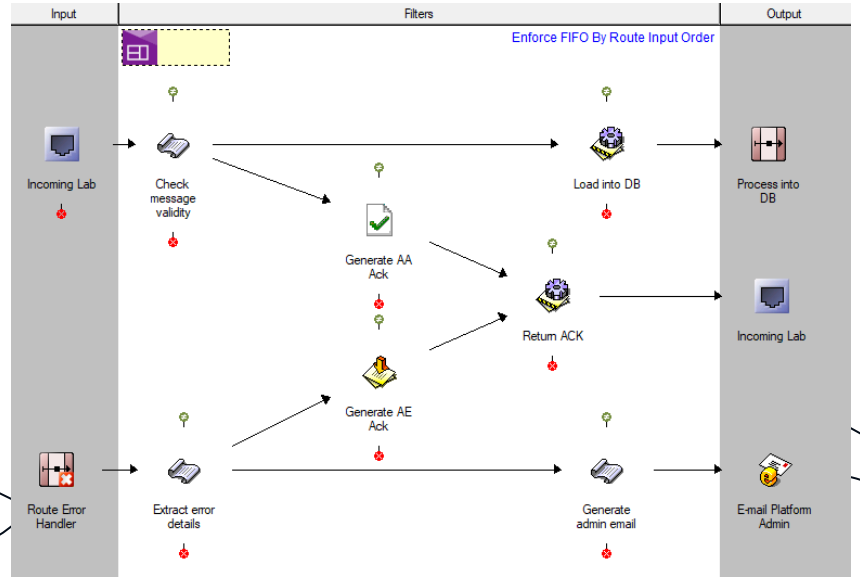
Principles

- Ensure error management is integral to your route design
- Test your configuration at regular intervals
- Include logic to in your solution to detect unknown messages

Tools

- Use the Conditional Connector
- Route Error Handler
- Message Collector Timeouts
- Continued Routing on Failure





LYNIATE

Inspecting Message Errors in JavaScript

- Use getErrors() function

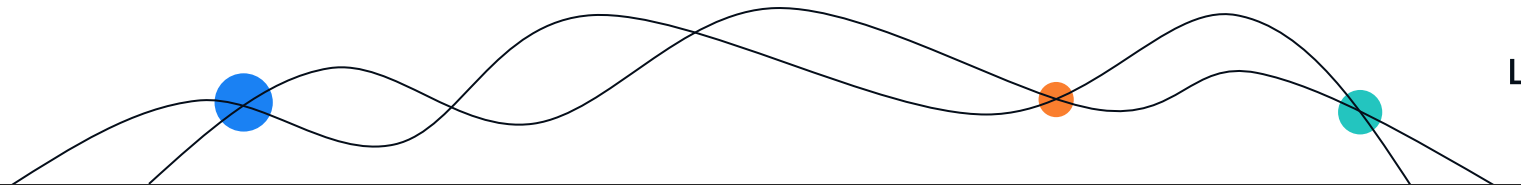
```
// Create the output message
var next = output.append(input[0]);

var errorText = "";
var errors = next.getErrors();

//Get all errors associated with message
if(errors != null && errors.length > 0) {
    for (var cErr = 0; cErr < errors.length; cErr++) {
        errorText = errorText + errors[cErr] + "\r\n";
    }
}

next.setProperty("ErrorMessage", errorText);
```

Testing

- Filter testing
- Conditional connector testing
- Test configuration:
  - Directory communication point
  - Timer communication point
  - TCP Communication points and route for mock system
  - EDI Explorer

Reconnect Settings

- Default behavior retries 5 times immediately then stops communication point
- Retry delay: Linear vs. Exponential
- Number of retries
- Retries per message
- Out->In and In->Out settings



Track Outbound Messages

- An important part of guaranteed delivery
- Methods: Synchronous & Asynchronous

Review & Questions