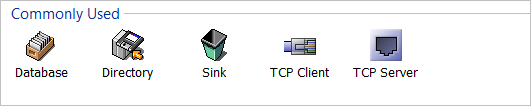
**Communication Points - Role and Functions**

**Communication Points**

Communication points are components which support receiving or sending messages. Each type of communication point implements a specific communication protocol and is responsible for receiving or sending messages using that protocol and for managing the transition of the message  to and from the route processing pipeline.

Rhapsody provides support for a wide range of communication protocols through the set of communication points supplied with the standard installation. These communication points share many common configuration properties as well as protocol specific parameters.

This module examines configuration properties common to the range of communication points and reviews the usage of commonly used communication  points:



The knowledge gained in this module and the following module (Filters) will be brought together in the Complex Route module (Ren-107) when we build and configure a more complex route than the examples so far.

### The Communication Points Dialog Box

Once a communication point is created, it is controlled by a set of properties accessible through the communication point properties dialog box. While some of the properties are specific to the protocol implemented by the communication point, many are common to all of the communication points. These will be examined in this module.

Rhapsody utilizes a common dialog format for all communication points which simplifies their management.

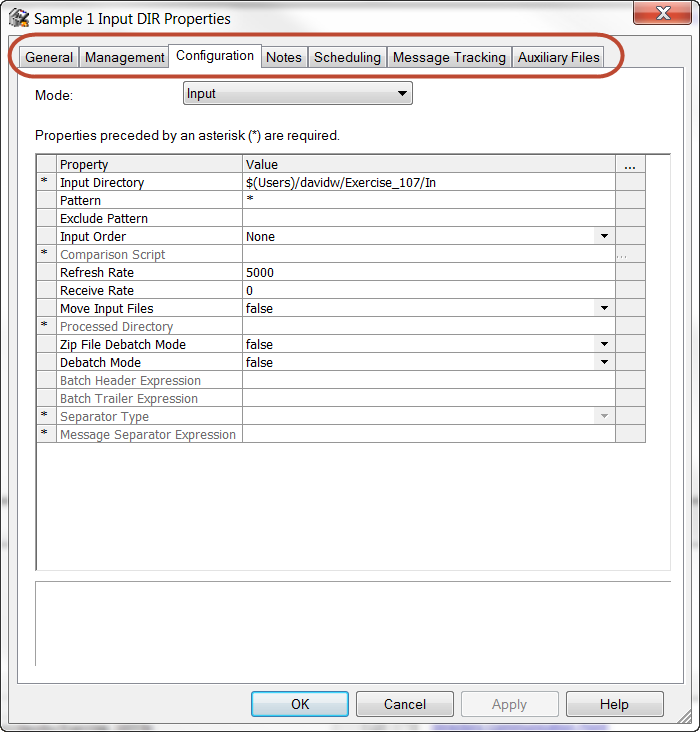
Access this dialog by right clicking on a **Communication Point** component in the **Configuration Workspace** window or **Route Canvas** and accessing **Properties** from its context menu.

The **Properties** dialog can also be opened by double clicking on the component.

The dialog groups the available functions using a tab format, covering the following function groups:

* **General:** general parameters including;
  + communication point name
  + direction of message flow
  + startup and retry behaviour
* **Management:** management of message rate.
* **Configuration:** protocol specific parameters.
* **Notes:** free text notepad to allow documentation of instance-specific details for a communication point.
* **Scheduling:** optional control of automatic start and stop for the communication point.
* **Message Tracking:** support for automatic matching of messages and their responses based on defined schemes.
* **Auxiliary Files:** provides the ability to attach additional drivers and control files specific to a communication point.

By default, the Communication Point Properties dialog opens at the **Configuration** tab.



**The General Tab**

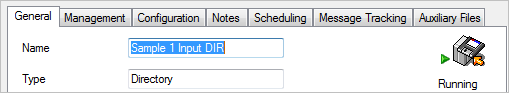
The **General** tab of the communication point dialog allows management of the communication point name and general properties. It includes the following sections:

* **General Properties**: direction and management of state at startup.
* **Connection Retries**: manage the behaviour of the communication point if the connection to a remote host fails.
* **Out->In and In->Out Properties**: manage the behaviour of message sequencing for the behaviour if the communication point is configured to support synchronous connections to remote hosts.

The **Name** section of the dialogue allows the user to change the name of the communication point and also displays the current status of the communication point.

An additional entry indicates the type of the component.

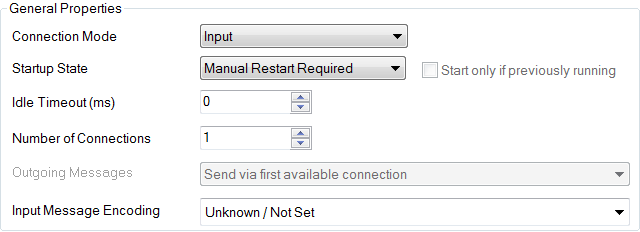
To change the name of the communication point, enter in the new name and then press the **Apply**button and **OK** to save.



**General Properties**

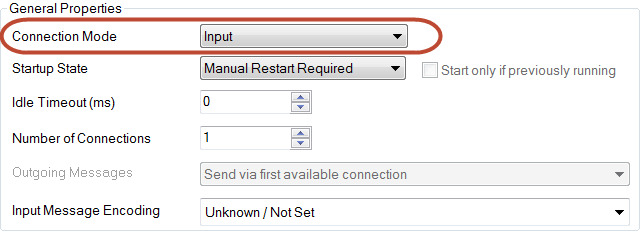
The General Properties are:

* **Connection Mode**
* **Startup State**
* **Idle Timeout**
* **Number of Connections**
* **Outgoing Messages**
* **Input Message Encoding**



**Connection Mode**

Rhapsody provides the ability for communication points to operate in a number of modes:



* **Input**: asynchronous, accepts inbound messages (can only be configured in the Input section of a route).
* **Output**: asynchronous, transmits outbound messages (can only be configured in the Output section of a route).
* **Bi-directional**: asynchronous, accepts input messages and transmits outbound messages as required (therefore must be configured in both the input section and the output section of a route).
* **In->Out**: synchronous, accepts an input message and then blocks input until the message has been sent out from the communication point (therefore must be configured in both the input section and the output section of a route).
* **Out->In**: synchronous, transmits a message and then blocks until it receives a response (therefore must be configured in both the input section and the output section of a route).   
    
  Note that the general behaviour of communication points in Out->In mode is to pass the message object (body and properties) through un-modified. The TCP communication points are an exception to this rule as they block until they receive a response from the remote host.

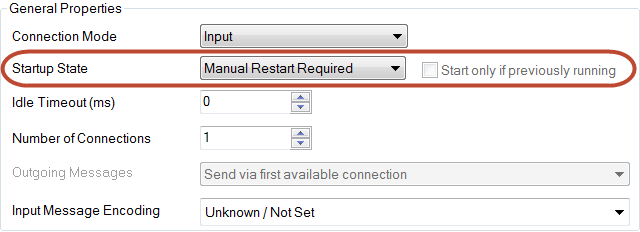
**Startup State**

Rhapsody provides a fine grained control of the order in which components start when the engine starts. The user can select from:

* **Manual startup:** communication points which are manually started as required.
* **Startup states 1 - 5:** control the order in which components start.

In the normal mode of operation, if a component is set for automatic startup, it will restart if the engine is restarted. Rhapsody will attempt to simultaneously start all components with the same startup state.

However, if a component has been stopped prior to an engine restart, it may not be desirable to have the component start automatically during the restart. This behaviour can be overriden to ensure the communication point is only started if it was running when the engine was stopped prior to the restart (see image below).



### Idle Timeout/Connections

### Idle Timeout

A communication point may be configured to stop automatically if it is inactive for a period of time. The default setting (**0**) is to remain in the running state even if there is no message activity.

### general_properties_4.png

### Number of Connections

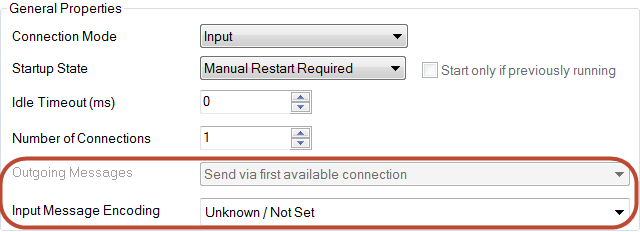
Most communication points can maintain an active connection with multiple systems at the same time (the exception is the Directory communication point).

If configured for multiple connections, Rhapsody identifies the connection on which the message is handled to assist with later processing if required.

### Messages

### Outgoing Messages

This item defines the way in which the communication point will handle errors when it attempts to further transmit a messages it has received (that is, when the Connection Mode is Bi-Directional or In->Out). The default transmission mode is to send the response message using the Input connection.  
  
This is particularly important when the communication point has multiple connections and is receiving input from multiple systems.



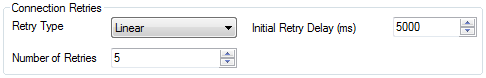
### Input Message Encoding

Rhapsody maintains a message attribute which identifies the character encoding system for a message.   
  
This ensures the engine is able to decode the body of the message correctly for manipulation and for viewing in the Management Console.   
  
The engine supports the encoding systems defined by the Java encodings.

The encoding attribute defaults to **UTF8**, but can be explicitly defined at the input communication point.   
  
If the encoding set is likely to vary on a message by message basis, the input communication point can be set to detect the encoding system.

**Connection Retries**

These parameters control the manner in which Rhapsody handles the failure of a connection to or from a remote system (that is a **physical connection**). In particular, whether an attempt should be made to reconnect and the frequency of attempts.



**Retry Type**

The Retry Type parameter may be set to:

* **No Retry**: do not attempt reconnection.
* **Immediate Retry**: attempt to reconnect immediately without any delay; if multiple retries are defined, these are executed without delay. Consequently, if the disconnection is due to external factors (for example a network outage), this mode is unlikely to be able to reconnect.
* **Linear Retry**: attempt to reconnect after a delay; if multiple retries are defined, use the same delay between each attempt.
* **Exponential Retry**: attempt reconnection after a delay, exponentially increasing the delay time between subsequent attempts.

**Initial Retry Delay**

This parameter establishes the delay before the first and subsequent reconnection attempts are made.   
  
Note that the manner in which the value is used depends on the Retry Type parameter.

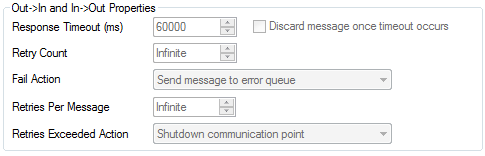
**Number of Retries**

This parameter defines the number of times to attempt the reconnection. Note that if the Retry Type is set to Immediate, the retry attempts are made very rapidly up to the Retry Count value.

The Number of Retries can be set to Infinite, allowing the connection to be re-established after long outages in most circumstances.

**Out->In and In->Out**

These configuration items control the management of messages when the communication point is set into a synchronous processing mode (to manage the connection at a **logical level**).



The behaviour of the communication point when set to **In->Out** mode is as follows:

* Receive a message.
* Block for Input.
* Process the message.
* Send a response back on (usually on the same connection).
* Unblock and wait for the next input message.

The behaviour of the communication point when set to **Out->In** mode is as follows:

* Sends a message.
* Blocks for receiving messages from the route.
* Waits for a response.
* Receives a response and places it on the processing queue.
* Enables message receipt.

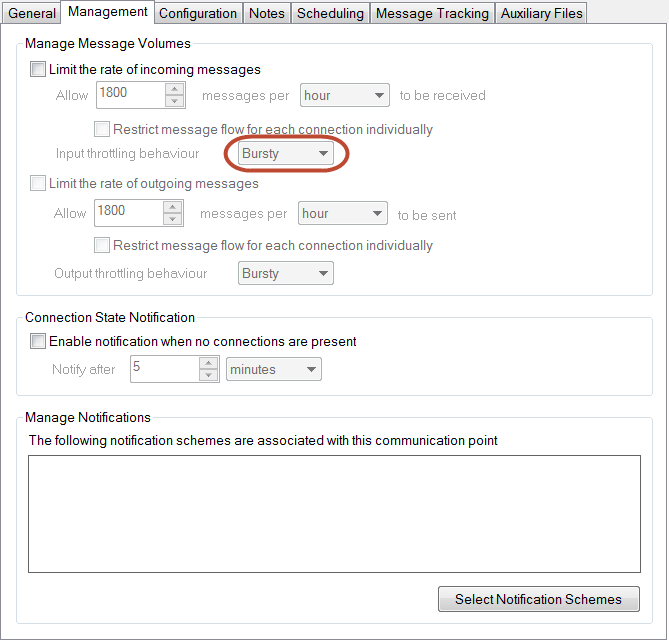
Note that when configured for synchronous processing, the communication point implicitly enables connection tracking (that is, the engine will attempt to use the same connection for output as the message was received on or for receipt on the same port as the original message was sent).

**The Management Tab**

The Management Tab provides a method of controlling the rate of receipt or sending of messages to cater for situations where processing queues back up for input messages or a downstream system is not able to accept messages as rapidly as the engine can send them.   
  
For input communication points, the rate controls the release of messages to the processing queue by the communication point (in other words, the messages are held at the input communication point and may cause the input queue to grow).   
  
For output communication points, the rate controls the release of messages to the output communication point, meaning the processing queue may back up.

The rate of input and output messages can be set independently, and can be applied either to message traffic through the communication point or through each connection independently.

Two modes of control are available:

* **Bursty**: the defined number of messages (message rate) are sent at the start of the interval; for example, if the rate is sent at 10 messages per hour, the first 10 messages are sent as soon as they are available.
* **Linear**: the defined number of messages are sent at equal intervals through the time period.  
    
  

The remaining configuration items on this tab provide administrative support for monitoring the state of the component and will not be covered in this course.

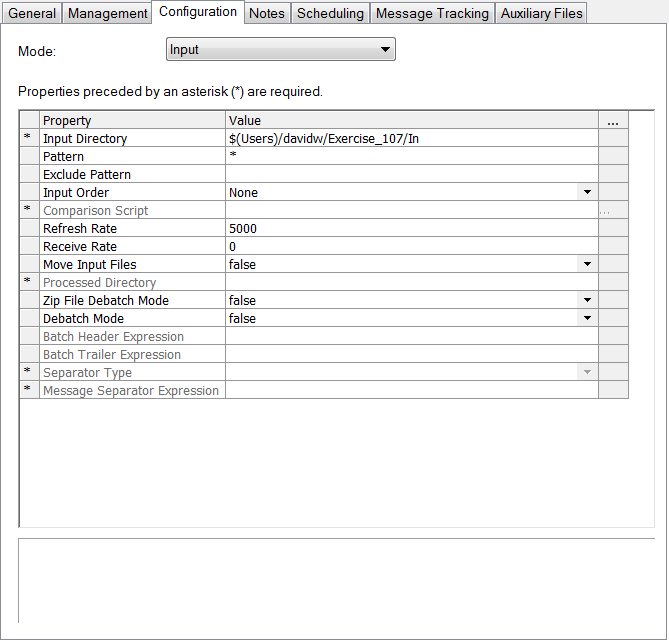
**The Configuration Tab**

The Configuration tab provides the protocol specific parameters for each communication point.

The parameters *required* for a communication point are indicated by an asterisk (**\***). Parameter values are either free form text or selected from a drop-down list of options. Free form text fields may be populated with:

* Text.
* The name of a message property, prefixed by **$** (for example, **$MessageType**).
* A Rhapsody Global Variable (often used to hold system specific values).

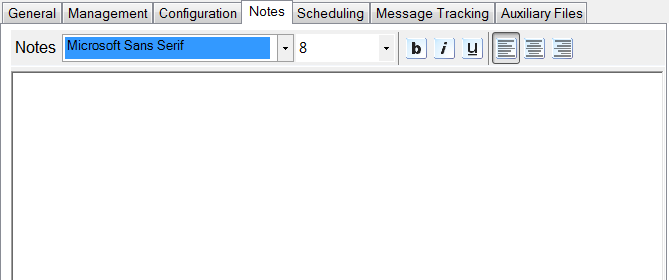
Each parameter provides a brief description of the purpose of the parameter.



### The Notes Tab

The Notes tab provides a free form text entry window allowing communication point documentation to be stored with the configuration.

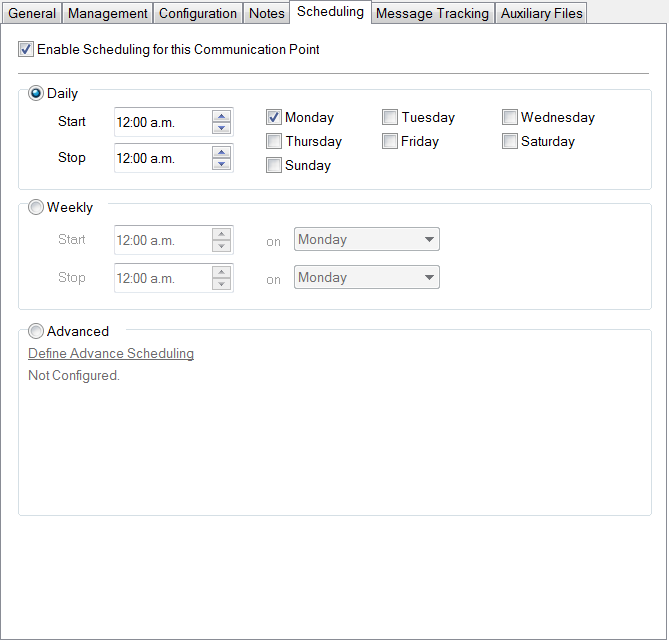
The IDE Documentation function includes notes stored in this tab with the system documentation.



### The Scheduling Tab

The Scheduling tab allows a fine grained schedule to be established where a communication point should only be active at specific times, for example, to process messages overnight rather than during the day, or to stop communication points to allow an external process to take place.

The schedule can be defined as a daily or weekly schedule using the dialogue, or a Unix cron like expression (time based job scheduler) can be entered using the Advanced parameter section. Note that the cron expression is extended to include a seconds parameter.



### The Message Tracking Tab

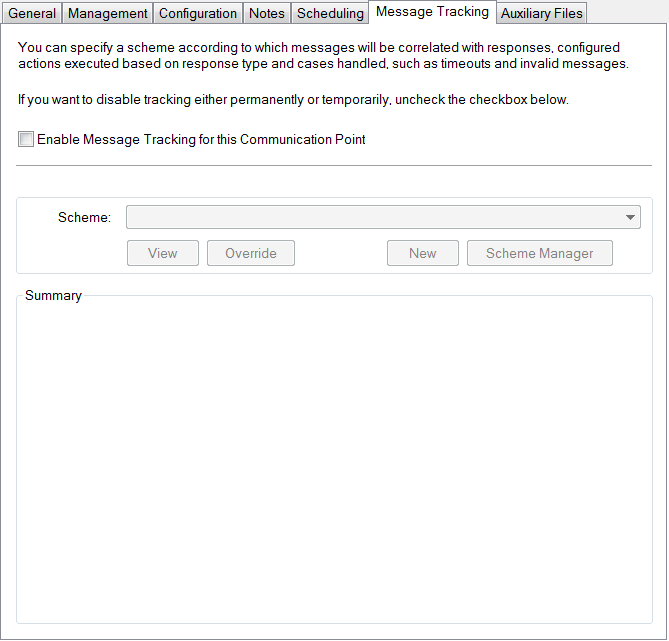
The **Message Tracking** functionality allows the engine to automatically detect a response to a message by utilizing a field value present in both messages.

The **Message Tracking** tab allows the user to define a tracking scheme based on a Symmphonia message definition to identify the fields in both the sent and received messages which will be monitored for a match.

The schemes also support management of retries if the response message is not received in a timely manner and also control of the behavior if an error occurs or no response is received and the retries are exhausted.

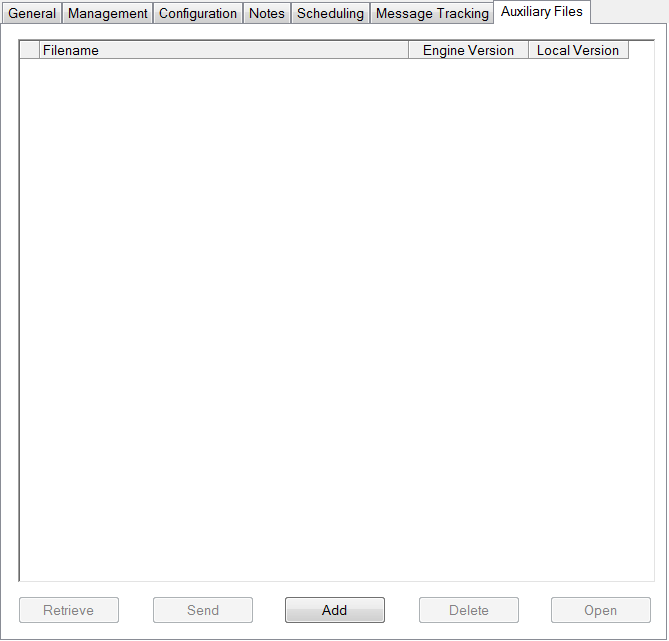
The engine allows multiple schemes to be defined.

This functionality is most frequently used in the HL7 messaging environment to manage the message/acknowledgement requirements.



### The Auxiliary Files Tab

The auxiliary files tab allows attaching additional support files to the communication point to provide the required functionality.   
  
The primary example of its use is the Database communication point where it is used to attach a SQL definition script and occasionally to attach a specific JDBC driver required for the connection to a specific database environment.



**Exercise 105A: Communication point properties**

**Attempt: 1**

**Overview**

This exercise identifies the common tasks required for configuring communication points and examines two key properties in detail (Retry management and Throughput management). The exercise uses the Directory communication point (introduced in Exercise 104) and introduces the TCP Client communication point.

The exercise requires the following steps:

* Create a new folder and route.
* Create and configure a Sink communication point and add it to the route.
* Create and configure a TCP client communication and add it to the route.
* Test the Retry management models with the TCP Client communication point.
* Create and configure a Directory communication point and add it to the route.
* Test the throughput management models with the Directory communication point.

**Creating the exercise environment**

Each communication point uses both common and protocol specific configuration properties (the latter presented on the Configuration tab of the dialog).

A number of common properties should be reviewed and updated as required for each new communication point, including;

* **General Tab**
  + Connection Mode (message direction); note that not all modes are supported by some communication point types.
  + Startup State; allows management of the order in which communication points are started at engine startup.
  + Connection Retries; defines the behavior required if connection issues occur
* **Management Tab**
  + Manage Message Volumes; provides support for limiting the rate at which messages are received or sent by the communication point.

This exercise will utilize a TCP Client communication point to investigate the management of retries and a Directory communication point to examine management of message volumes.

Create a new folder in your User folder for this exercise and create a route and a Sink communication point within the new folder.

**Properties of the Sink**

The Sink communication point provides the ability to cleanly discard unwanted messages and is used in this exercise to provide a convenient output.

Configuration for a Sink communication point is minimal as it is always an output component and it has no protocol specific properties.

**Managing Startup State**

The **Startup State** should always be set for this component on the General tab of the configuration dialog to ensure that it is running as required after an engine restart.

When the engine starts, it supports a sequential start of up to five groups of components (each component in a group is effectively started simultaneously) which ensures that resources are available as required, and that minimal queueing of messages occurs during the startup process.

A **Manual Restart** state is also provided for components which are intermittently started or which are not required to run all the time and are automatically started on a regular schedule.

Select a suitable startup state for the sink component and exit from the configuration dialog.

**Retry Management - the TCP Client**

The TCP communication points are used to transfer messages over a network using the TCP protocol. The **Client** provides the ability to create a connection to a remote host over which the messages are transferred, while the **Server** provides a listener to support connections made by a remote host.

**Exercise Tasks**

This exercise uses the TCP Client to examine the retry behavior of the communication point when the connection **cannot** be established at a **network** level.

1. Create a new folder in the IDE for this exercise. Right click the folder and, from its context menu, select **New Route**. Give the new route a name; for example Ex105a.
2. Create a new TCP Client communication point in the folder.

**General Tab**

1. Review the **General tab** of the **Configuration** dialog for the communication point and set the following properties:
   * **Connection Mode**: Output
   * **Startup State**: as desired (note that this setting only applies to management of the state when the engine starts).

**Configuration Tab**

1. Review the **Configuration Tab** properties.
   * This tab provides the ability to define the connection details for the host to which we wish to connect (**Remote Host** and **Remote Port**) as well as parameters about the connection.

As this exercise mimics an error in the connection to observe the retry state, we will use the following settings:

* + **Remote Host**: localhost   
    (that is, the communication point will make connection attempts to the server the engine is running on).
  + **Remote Port**: 60001   
    (60001 is a network port which is not normally used by any application; consequently we expect the connection to fail).

The rest of the parameters may be left as the default settings.

1. Drag the **TCP Client** communication point onto the **Output** portion of the route. (Note that for this exercise a connector is not required to the output communication point as we will not send any messages to it).



1. Check in and start the route. (The TCP Client communication point will be started at the next step).

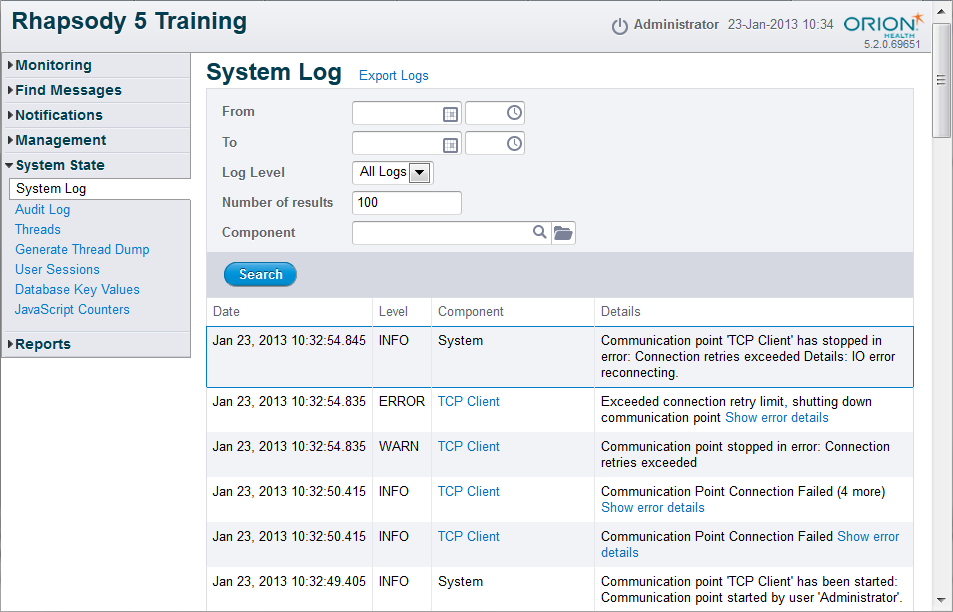
**Immediate Retry Behavior**

The communication point **General** tab supports three models to control retry (or re-connection) behavior if a connection is broken (**network level**):

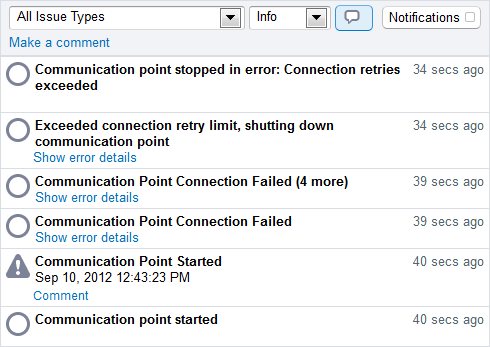
* **Immediate**  
    
  If the connection fails, immediately attempt to re-connect; repeat for the number of attempts defined by the **Number of Retries** parameter (note that the **Initial Retry Delay** is ignored in this mode).
* **Linear**  
    
  If the connection fails, wait for the period defined by the **Initial Retry Delay** and attempt to re-connect; repeat for the number of attempts defined by the **Number of Retries** parameter.
* **Exponential**  
    
  If the connection fails, wait for the period defined by the **Initial Retry Delay** and attempt to re-connect; repeat for the number of attempts defined by the **Number of Retries** parameter, extending the retry interval exponentially, using the formula:  
    
  *Initial Retry Delay (ms) x 2(retry number-1)*

**Exercise Tasks (Immediate retry)**

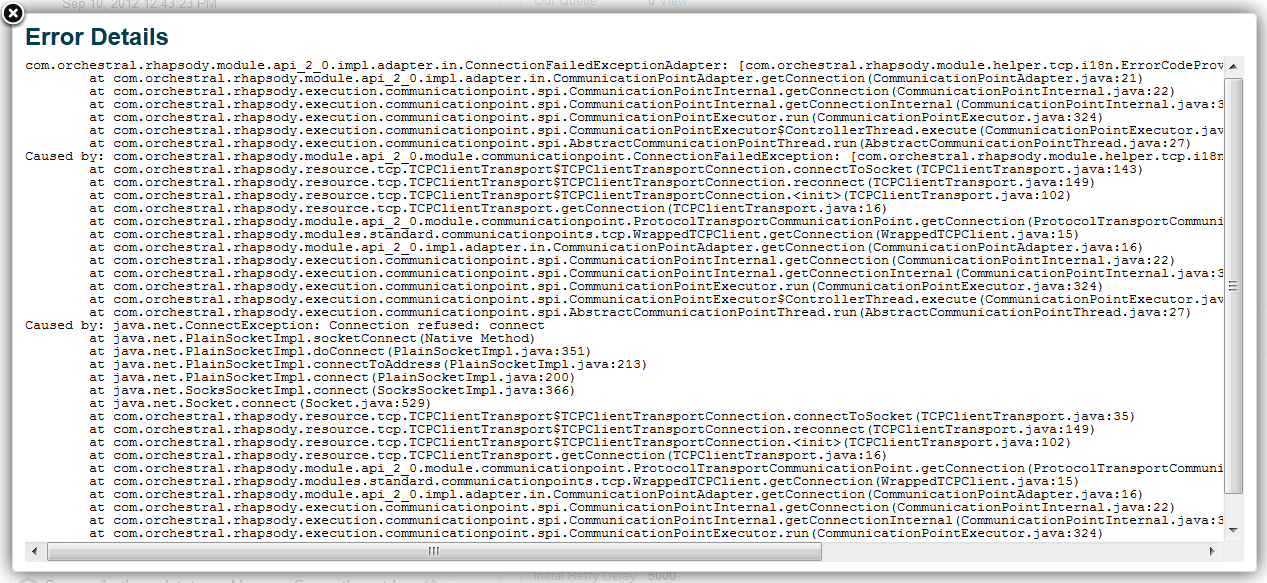
1. Configure the following in the **General** tab of the **TCP Client** communication point.
   * **Retry Type**: Immediate
   * **Number of Retries**: 5
2. Check in the changes and start the communication point. It will eventually fail (after 5 retries) and shut down. This can be seen by the red exclamation mark beside the communication point in the **Configuration Workspace**.
3. Open the **Web Management Console** and examine the **System Log**.  
     
   The **System Log** page will display the attempt to **start** the communication point, the **initial failure** followed by the retries, and the subsequent **stopping** of the component (when the connection retry limit has been exceeded). The timestamp of the events is shown in seconds (with milliseconds shown after the decimal point).



1. Review the time stamps for the sequence and note the time interval between **starting** the communication point, the time delay recorded for the **initial connection failure** and the time delay recorded for **shutting down** the communication point.
2. **Note down the results**. These values will be required in a later question page.
3. Navigate to the details page for the **TCP Client** communication point and examine the log window (you may need to select/toggle the **Activity** link if there are any active alerts). The image below is what you expect to see:



1. Open the **Show error details** link on the **Communication Point Connection Failed** line and observe the error detail. Examining the **Caused by** detail should help identify the error.
   * For example **Caused by**:   
       
     com.orchestral.rhapsody.module.api\_2\_0.module.communicationpoint.ConnectionFailedException: [com.orchestral.rhapsody.module.helper.tcp.i18n.ErrorCodeProvider/\_491306573\_5104] IO error reconnecting



**Linear and Exponential Retry Behavior**

**Exercise Tasks (Linear Retry)**

1. Re-configure the **Connection Retries** parameters as follows:
   * **Retry Type**: **Linear**
   * **Initial Retry Delay**: **5000**  
     note that this parameter is expressed in milli-seconds
   * **Number of Retries**: **5**
2. Check in the changes and start the communication point. It will eventually fail (after 5 retries) and shut down.
3. Open the **Web Management Console** and examine the **System Log**.
4. Review the time stamps for the sequence and note the time interval between **starting** the communication point, the time delay recorded for the **initial connection failure** and the time delay recorded for **shutting down** the communication point.
5. **Note down the results**. These values will be required in a later question page.

**Exercise Tasks (Exponential Retry)**

1. Re-configure the Connection Retries parameters as follows:
   * **Retry Type**: **Exponential**
   * **Initial Retry Delay**: **5000**
   * **Number of Retries**: **5**
2. Check in the changes and start the communication point. It will eventually fail (after 5 retries) and shut down.
3. Open the **Web Management Console** and examine the **System Log**.
4. Review the time stamps for the sequence and note the time interval between **starting** the communication point, the time delay recorded for the **initial connection failure** and the time delay recorded for **shutting down** the communication point.
5. **Note down the results**. These values will be required in a later question page

**Throughput Management - the Directory Communication Point**

The **Management** tab of the configuration dialog allows limiting the rate at which messages are handled by the communication point.

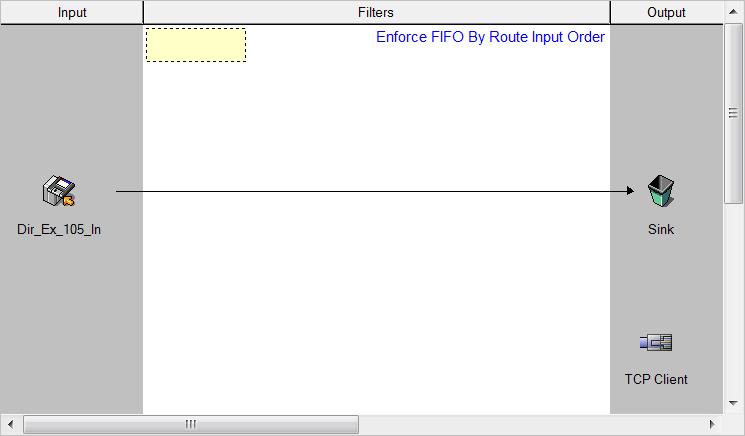
This is frequently necessary where either a route or a downstream system is not able to process messaging sufficiently rapidly to handle the load, resulting in queueing in the system at best, or, at worst, loss of messaging by an external system.

**Exercise Tasks**

1. Create a **Directory** communication point to use for this exercise with the following configuration:
   * **Connection Mode**: **Input**
   * **Input Directory**: path to a folder in **/rhapsody/Users/<your\_user\_name>**

The rate at which messages may be handled by the communication point is defined by the number of messages permitted per time interval.

1. On the **Management** tab, select **Limit the rate of incoming messages:**
   * **Allow**: **5 messages**
   * **Per**: **minute**
2. Create/modify the route as shown below:



### Message Throttling

### Input Throttling Behavior

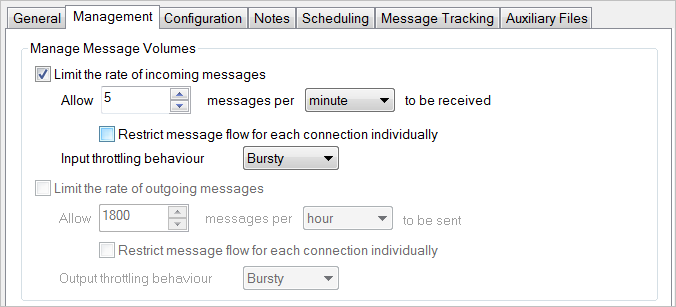
The rate limiting functionality supports two modes of controlling the flow of messaging through the communication point: bursty and linear.

The **Bursty** mode maintains the throttling model of earlier Rhapsody versions which transmits the messages as a group at the start of the time interval. This results in a rather spiked pattern of message transmission.

The **Linear** mode spreads the message transmission linearly across the transmission time interval, resulting in a smooth pattern of message transmission.

### Exercise Tasks

#### Bursty Behavior



1. Configure the **Directory** communication point to use **Bursty** behavior (the default).
2. Check in the communication point and start it.
3. Copy the complete set of ten test messages for this exercise and paste them into the input directory for the communication point.
4. Monitor the directory, refreshing it, and note that the directory does not clear immediately (that is, only the first group of messages are handled by the communication point at startup).
5. Review the **Throughput** graph for the Directory communication point in the **Management Console** and examine the message transmission pattern.  
     
   You may need to refresh the graph until the messages have been fully processed.

#### Linear Behavior

Re-configure the communication point to use **linear throttling behavior** and repeat the above steps.

### 105A Model Answer

### What you expect to see:

