Tracker iRouter User Guide

John Conlon and contributors

Chapter 1. Overview

Document Status

This document is the second draft of an iRouter User Guide and is only intended as an overview to the technologies used in the Tracker iRouter. Future versions will fully serve the intended function of a comprehensive user guide for the installation and configuration of the iRouter.

Information Modeling and Processing

Custom integration solutions for real time physical sensor and actuator data are complex, error prone and time consuming.

Can a generic solution be found for the acquisition, modeling, processing and exchange of physical measurement and state information? One that leverages industry standards and reflects current software engineering best practices? One that hides complexity from end user data acquisition operators? One that can run on a desktop or in a headless appliance?

Solution Requirements

In order to provide a generic solution to problems dealing with the sensing and processing of physical measurements and physical states, a comprehensive solution will require technology for:

1. Modeling measurement and state data [1] [2]
2. Creating services for:
   - Interfacing with external devices for data acquisition from physical sensors
   - Data transformation and computation between services
   - Interfacing with external devices for the output of measurement and state data to physical actuators
3. Creating the interconnections between compatible services
4. Managing, persisting, importing and exporting the configuration data for service components and interconnections
5. Orchestrating the establishment of connections between compatible Producer and Consumer services
6. For operators running the tool:
   - Simplify the operation by hiding the complexity of all of the above
7. For administrators needing configuration validation and troubleshooting aids, provide monitoring and diagnostic tools to visualize:
   - Configuration
   - Internal operation of the services

Leveraging Industry Standards

The OSGi Alliance offers open specifications that enable the modular assembly of software built with Java technology. These OSGi specification meet most of the solution requirements outlined above.

The following sections overview the pertinent technologies specified by the OSGi and their relationship to the above solution requirements. [2] [2]

OSGi Measurement and State

The OSGi Measurement and State specification offers a comprehensive [3] [2] way to model physical measurement and state data.

Meets requirement #1.

Wire Admin

A second OSGi specification called the Wire Admin Service specifies how to:

- Represent services which:
  - Produce measurement and state information
  - Consumer measurement and state information
- Represent wire connections that move measurement and state information between producer and consumer services

Meets requirements #2 and #3.

Configuration Admin and Metatype

Lastly two more OSGi specifications provide the required solutions for configuration administration:

- OSGi Metatype Service
- OSGi Configuration Admin Service

Meets requirement #4.

What is the Tracker iRouter?

The Tracker *Information Router*, **iRouter** is a generic integration solution for acquiring, transforming and exchanging information between network connected sensors and actuators.

The Tracker iRouter complements OSGi standard services by adding:

- An *orchestration* framework that hides the complexity of establishing the connections between OSGi WireAdmin services
- User interfaces for simplifying configuration management and application monitoring

The Tracker iRouter can run as a headless/keyboardless network appliance or as a background service in an Eclipse based desktop.

iRouter – A Just-In-Time integration solution for multiple input devices, sensors and actuators.

Notes

[1] The majority of information moving between components in the iRouter will be Measurement and State data.

[2] Implementations of these specifications are available from several OSGi providers within the Open Source community.

[3] Because an OSGi Measurement can represent all SI units of measurement and record the precision of the measuring device, it can be used to model the complete range of measurements and the accuracy of the measuring devices.

Chapter 2. Theory of Operations

How does it work?

The iRouter is a running network of OSGi services that move and transform information between external sensors and actuators. An iRouter service network consists of Custom Proxy components that interface with external devices and internal components that execute business logic.

Custom proxy components for external devices

External measurement sensors, user input devices, and actuators are interfaced to the iRouter system with a custom component called a proxy and implemented as a WireAdmin Producer or Consumer service.

Interfacing to the external device requires that the proxy component:

1. Provide a physical communication interface
2. Provide a data link communication interface

For the end user or business analyst interfacing devices supported by available proxy components nothing more needs to be done other than configuration. But when there are requirements for integrating new external devices, new proxy components will have to be created by a qualified programmer.

Physical communication interfaces

Physical interfaces for all standard communication interfaces are already part of the existing iRouter functionality and do not have to be a concern to a programmer when creating a proxy. Physical interfaces are simply specified as connection *URI*s in the configuration.

The current iRouter support the following connections:

- Serial communication port connections
- Internet TCP socket connections
- Bluetooth wireless connections

Datalink communication interfaces

To the programmer authoring a proxy, the lower level physical connection is taken care of and all connections are presented as input and output streams. Only the processing of these streams at the higher level of the data link protocol is necessary. Therefore the complexity of a proxy component is directly related to the complexity of the data link protocol offered by the external device and the number of input or output measurements and states produced or consumed by the external device.

Simplified interfaces

While programing a proxy is still a custom *integration* effort, it is an order of magnitude easier typical custom programming efforts because:

- No low level connection programming is necessary
  - iRouter already has this covered
- No business logic needs to be coded
  - Business logic is handled by configuring transformer services – see next section
- Implementing a Proxy Producer or Consumer for an external device is only done once
  - All use cases that require the device reuse the same proxy

Generic transformation components

iRouter transformation components listed below provide the services for transforming measurements and states flowing within the iRouter. [4] [5] In other words they provide the business logic for each use-case.

Measurement Converters

Measurement converters convert one measurements to another based on an arithmetic calculation or some other relationship.

The following Measurement Converters are included:

- Simple Arithmetic Converter
  - Consumes one measurement input
  - Produces a measurement from the input and an arithmetic calculation

State Detectors

State detector components produce a state based on some criteria. The following state detectors components are included:

- Measurement Comparator
  - State is produced based on the comparison of two measurements.
    - Produces a state with a negative integer, zero, or a positive integer if the first measurement is less than, equal to, or greater than the second measurement.
    - If the states are unequal or incompatible creates a state with an Integer.MIN value.

Collaborating wire groups

Sensor-actuator use cases are actualized by a group of WireAdmin services interconnected with WireAdmin wires. These are called iRouter *wire groups*. Because the iRouter can support one or more sensor-actuator use cases or *wire groups* running at the same time, during configuration time each service must specify a name for the *wire group* it collaborates with.

All component services must be identified with a wire group name so that the iRouter can wire the correct groups of services together.

Administration Tools

Tools needed for managing the components running in the iRouter include:

- User interface tools for Configuration editing
- User interface tools for Application Monitoring

Configuration editing

Creating a running instance of the iRouter is simply a matter of creating, editing and saving a configuration. This is done through the iRouter configuration editor. With the iRouter configuration editor the administrator can:

- Select the type of component instances to create from a list of the deployed component factories,
- Configure component instances to perform specific functional tasks related to information acquisition and transformation
- Name the *wire groups* of the component instances to constrain the behavior of the instance to a set of collaborating services
- Add or delete component instances
- Edit the configuration of running component instances
- Import or export configuration data

When a configuration is saved all components are activated. Configurations are persisted across reboots, so turning on the iRouter appliance or re-launching the Eclipse desktop will activate all configured collaborating *wire group* services previously configured.

Application monitoring

Once a iRouter configuration is created and saved it is activated as a background set of running services. Administrators may want to check that these services are indeed collaborating in the manner originally imaged. With The iRouter monitor view administrators can:

- Graphically visualized the *wire groups* of running components
- Resize or reposition nodes on the graph to clarify the visualization
- Print the output of the visualization as a visual record of the configuration
- Monitor status variables of individual service components.

Simplified operation

Once the configuration is edited and saved by an administrator, the user no longer need perform any additional tasks - *because the iRouter is configured and running as a background service.*

Notes

[4] Initial iRouter beta releases will deploy only a limited number of generic components to the software repositories for provisioning iRouter instances. Follow up releases will add more generic transformation components, so for most use cases custom business logic components will not be needed.

Chapter 3. Use Cases

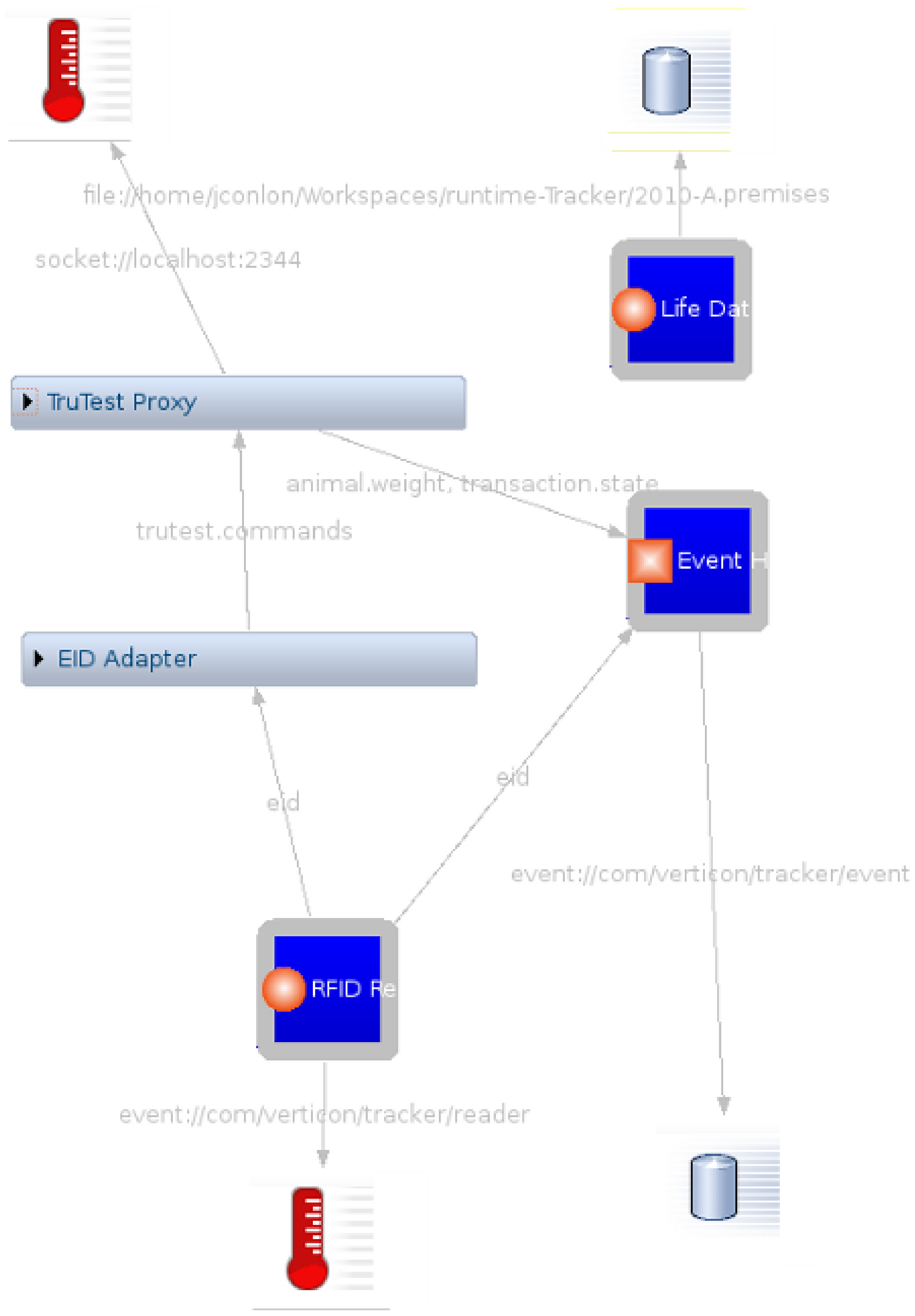
The following are some example use cases. These are only examples are in no way the only use cases possible, as the iRouter components can be mixed and matched to create many other custom configurations. Also note that the iRouter can run multiple wiring groups. This means that single or multiple instances of these use cases can be combined in one configuration and all run at the same time.

Premises Desktop Integration with Sensors

Intention

1. *Record animal weight measurement events from a TruTest scale.*
2. *Record measurement events sent to an operator running a Tracker Desktop transaction editing session.*
3. *Use one or more RFID readers to identify the animal EID tags.*

Monitor View


file://home/jconlon/Workspaces/runtime-Tracker/2010-A.premises
socket://localhost:2344
Life Dat
TruTest Proxy
animal.weight, transaction.state
trutest.commands
Event H
EID Adapter
eid
eid
event://com/verticon/tracker/event
RFID Re
event://com/verticon/tracker/reader

iRouter monitor view image capture of an integrated Premises Desktop

The above is the iRouter monitor view of a TruTest scale head integrated with user managed RFID readers within a Tracker Desktop transaction processing environment.

Configuration

The configuration includes:

- A TruTest proxy component for connecting to an external TruTest sensor/actuator *(A livestock weighing scale head with life data synchronization, eid display, and key entry)*
- A Life Data gateway component for synchronizing data from a specified Tracker Desktop Premises document to the TruTest scalehead.
- A EID gateway component for receiving Electronic Identification (EID) numbers from Tracker Desktop user managed RFID readers.
- A TruTest EID adapter that converts EID numbers to TruTest protocol commands.
- A Measurement gateway component for sending completed TruTest Livestock weight measurement transactions to the active Tracker Desktop transaction editor session.

Scenario

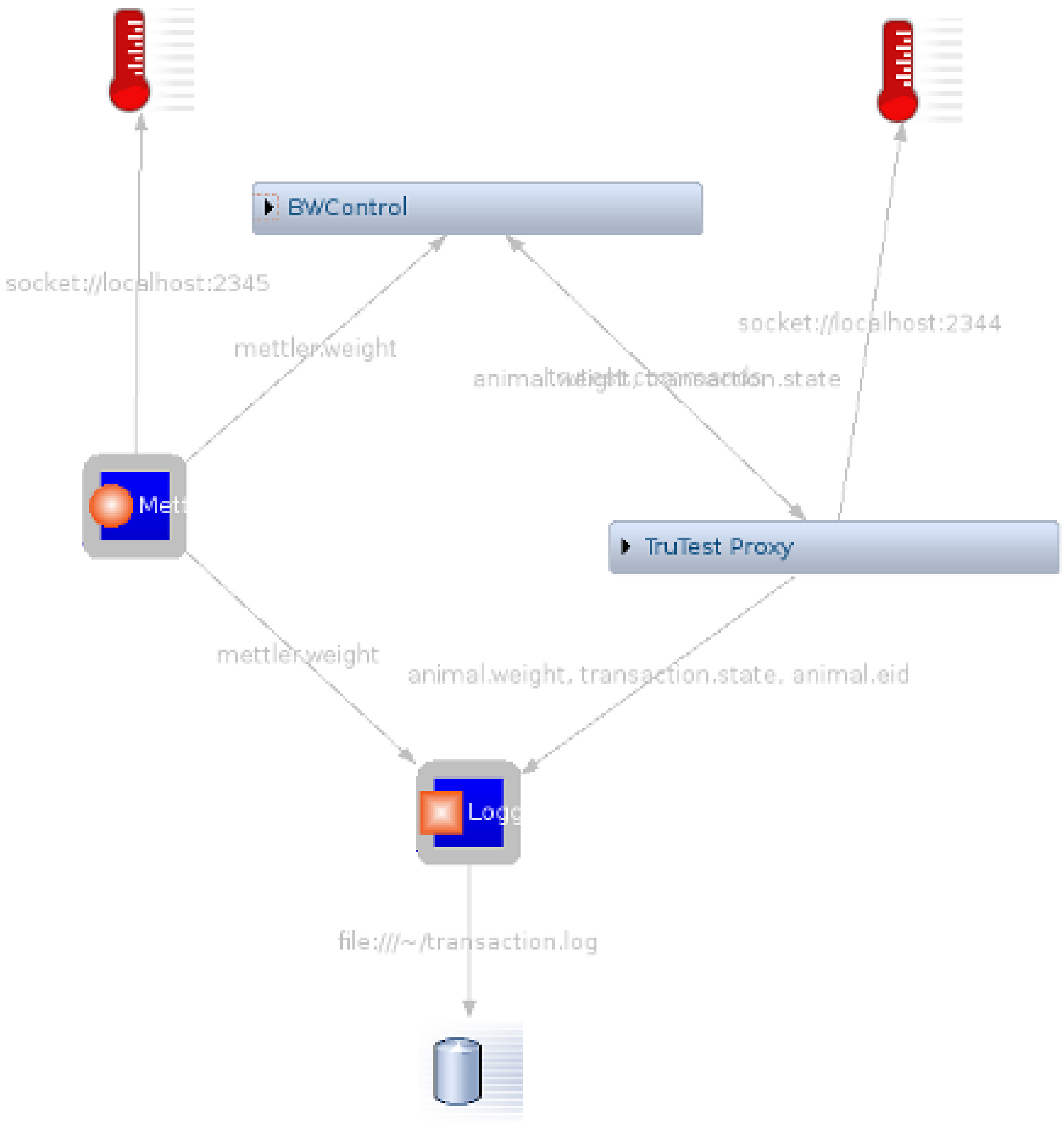
As the configuration services activate, the current animal life data will be extracted from the specified Animal Premises document and loaded into the TruTest scale head. Operators will use one or more RFID readers to read animal EID tags as the animal enter the weight stations. As a result the EID numbers will show up on the TruTest indicator. Animals will be weighed with the operator terminating the weight transaction with a press of the enter key. The final result of this transaction will be sent to active transaction session on the Tracker Desktop as a weight measurement event. This event along with any other events assigned to the Tag ID reader will be added to the document and will be seen by the Desktop operator as a new time stamped weight measurement event arriving in the Event View.

Headless Recording of Animal and Blood Weights to a Log File

Intention

1. *Log aggregated animal weight and blood weight measurement events from a TruTest scale and a Mettler laboratory balance.*
2. *Show blood weight target and dynamic readings on the TruTest scale indicator.*
3. *Send TruTest alarms to the operator when target blood weight is approached.*
4. *Use a headless appliance.*

Monitor View



iRouter monitor view image capture of an integrated multi-sensor data logger

The above is the iRouter monitor view of a TruTest scale head integrated with a Mettler laboratory scale and a data logger.

Configuration

While it is possible to use multiple generic components for business logic, complex business logic can instead be encapsulated in a single custom business logic component. This use case depicts using a single business logic component for integrating, controlling and transforming complex information network flows between multiple sensors and a data logger.

The configuration includes:

- Two custom proxy components for connecting to:
  - An external TruTest sensor/actuator *(with display, alarming, and key entry)*
  - An external Mettler balance weighing sensor
- One general purpose gateway for logging
- A custom business logic component for the information transformation and exchange

Scenario

Animal weight measurements, identification numbers and record state are sent from the TruTest producer on the TruTest proxy, animal fluid weight measurements are sent from the Mettler balance, the business logic component will do measurement transformation, measurement threshold detection and alarm signaling. The Measurement Logger consumer service writes to the logging backend. [5] [10]

Notes

[5] iRouter uses the enterprise class logging framework logback

Chapter 4. Getting Started

Getting Started Overview

iRouter is packaged as a set of software artifacts called features. These features must run in an Eclipse *RCP* running on the user desktop. The Tracker Business Intelligence Toolkit is just such a platform.

To get started:

1. Download the Tracker BI Toolkit Desktop and extract the archive
2. Start the application
3. Provision the Tracker Desktop with the iRouter features
   a. Go to the *Help* menu and choose *Install New Software...*
4. In the *Work With* choice box, choose the *Tracker Releases* repositories
   a. Select the iRouter features to install

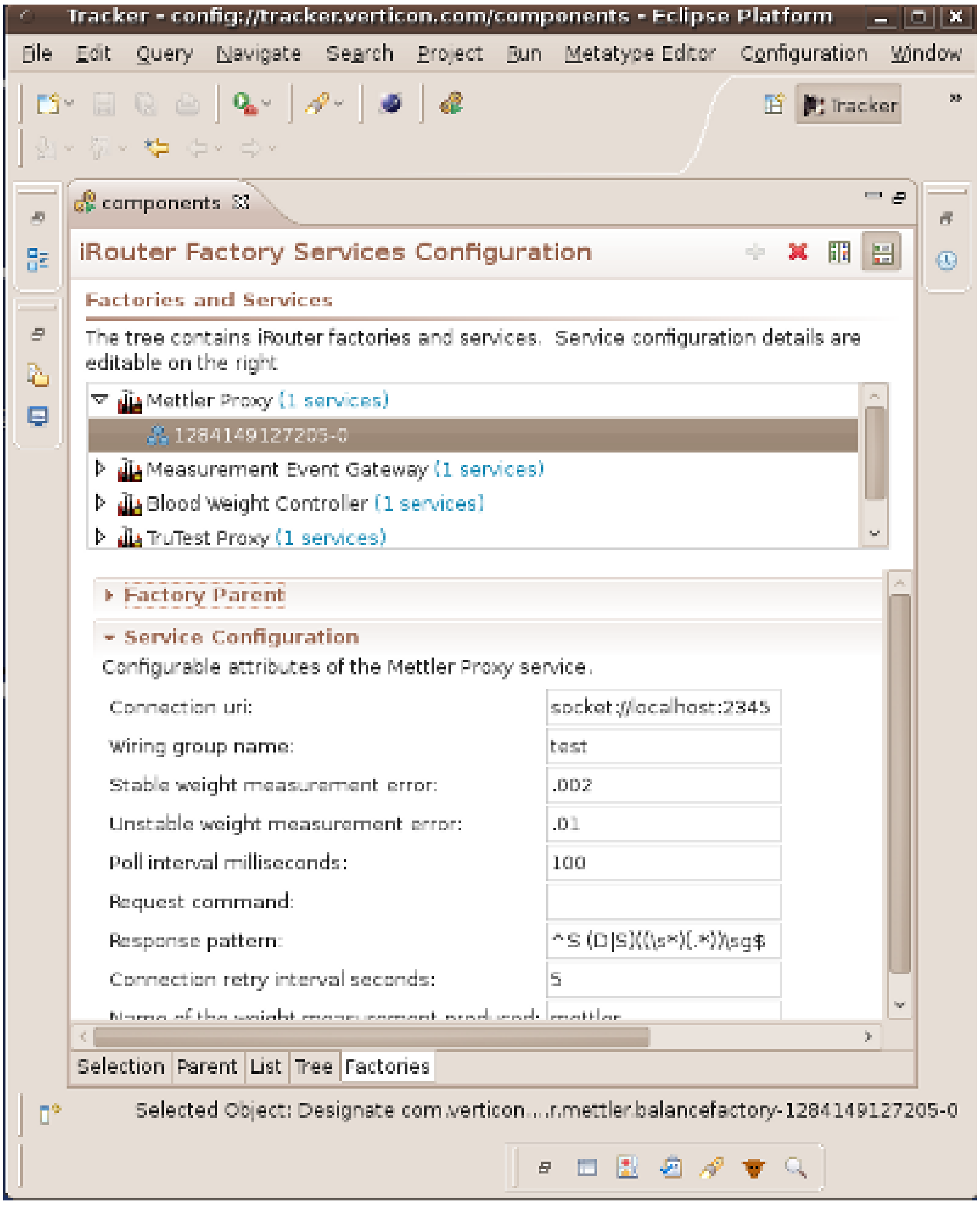
Chapter 5. Configuration

Configuration Administration Overview

iRouter problem solutions always begin by a rational examination of the problem, clarification of the use-case, and a theoretical design of a solution. The solution is then realized by adding the component instances representing the solution from a list of deployed component factories. Lastly the administrator configures the added instances and saves the configuration.

iRouter configurations conform to the OSGi **Metatype** document schema. One approach for editing configuration documents is to use an external *XML* (or text) editor to create the configuration and import it into the iRouter to activate it. While that may work for the more technical administrator, a far easier approach is to use the Tracker iRouter *Configuration Editor* for the editing process.

Configuration Editor


Tracker - config://tracker.verticon.com/components - Eclipse Platform
File Edit Query Navigate Search Project Run Metatype Editor Configuration Window
Tracker
components
iRouter Factory Services Configuration
Factories and Services

The tree contains iRouter factories and services. Service configuration details are editable on the right

Mettler Proxy (1 services)
1284149127205-0
Measurement Event Gateway (1 services)
Blood Weight Controller (1 services)
TruTest Proxy (1 services)

Factory Parent
Service Configuration

Configurable attributes of the Mettler Proxy service.

Connection uri:	socket://localhost:2345
Wiring group name:	test
Stable weight measurement error:	.002
Unstable weight measurement error:	.01
Poll interval milliseconds:	100
Request command:	
Response pattern:	^S (D|S)((\s*)(.*))\sg$
Connection retry interval seconds:	5
Name of the weight measurement producer:	mettler

Selected Object: Designate com.verticon....r.mettler.balancefactory-1284149127205-0

Example of the Configuration editor

iRouter Configuration Editor is a full featured *EMF* editor with a special *Factories* view 6 [13] that is optimized for dealing with iRouter Components components that have published metatype resources. The Factories view will be described below.

Configuration editor Factories view tab

Clicking on the *Factories* tab of the editor will bring up a master-detail page form. The master page will be a tree and the nodes of the tree will be all the iRouter Component Factories 7 [13] that where deployed 8 [13] to the iRouter.

Component Factory nodes in the tree will have children nodes called Service Instances 9 [13] .

Clicking on a Factory parent or a Service Instance child node in the master tree will show the details of the node in the details page 9 [13] .

Creating a service instance

Select the desired *Component Factory* in the master tree, from which you would like to create a service instance. The description of the *Component Factory* will show in the *Factory Properties* panel in the details page.

Select the green plus key in the upper right of the *Factories* editing window to instruct the factory to create a Service Instance. It will show up as an unconfigured and unregistered child of the Component Factory.

Editing a service instance

To edit the configuration of the Service Instance, select it in the master tree and edit the *Service Configuration* panel in the details page.

Saving the configuration

To save the new or a changed configuration select *File* and *Save*.

Importing and exporting configurations

To export the active configuration, select the *Configuration* menu and _Export iRouter Configuration_ and choose a directory and file name for the exported configuration file.

Likewise to import a new configuration, select the *Configuration* menu and _Import iRouter Configuration_ and choose a metatype file from the filesystem to import.

Notes

6 Future iRouter releases will include a more graphic editing facility.

7 Component factory is a *Factory for creating one or more service instances.*

8 All iRouter components are Component Factories. Like the Tracker Desktop, Tracker iRouter components are deployed and updated through the Equinox P2 provisioning infrastructure.

9 Service instance are the services that do the real work of moving and transforming data.

10 The orientation of the master-details pages can be reset to above and below. To change orientation use the orientation tools in the upper right of the view.

Chapter 6. Monitoring

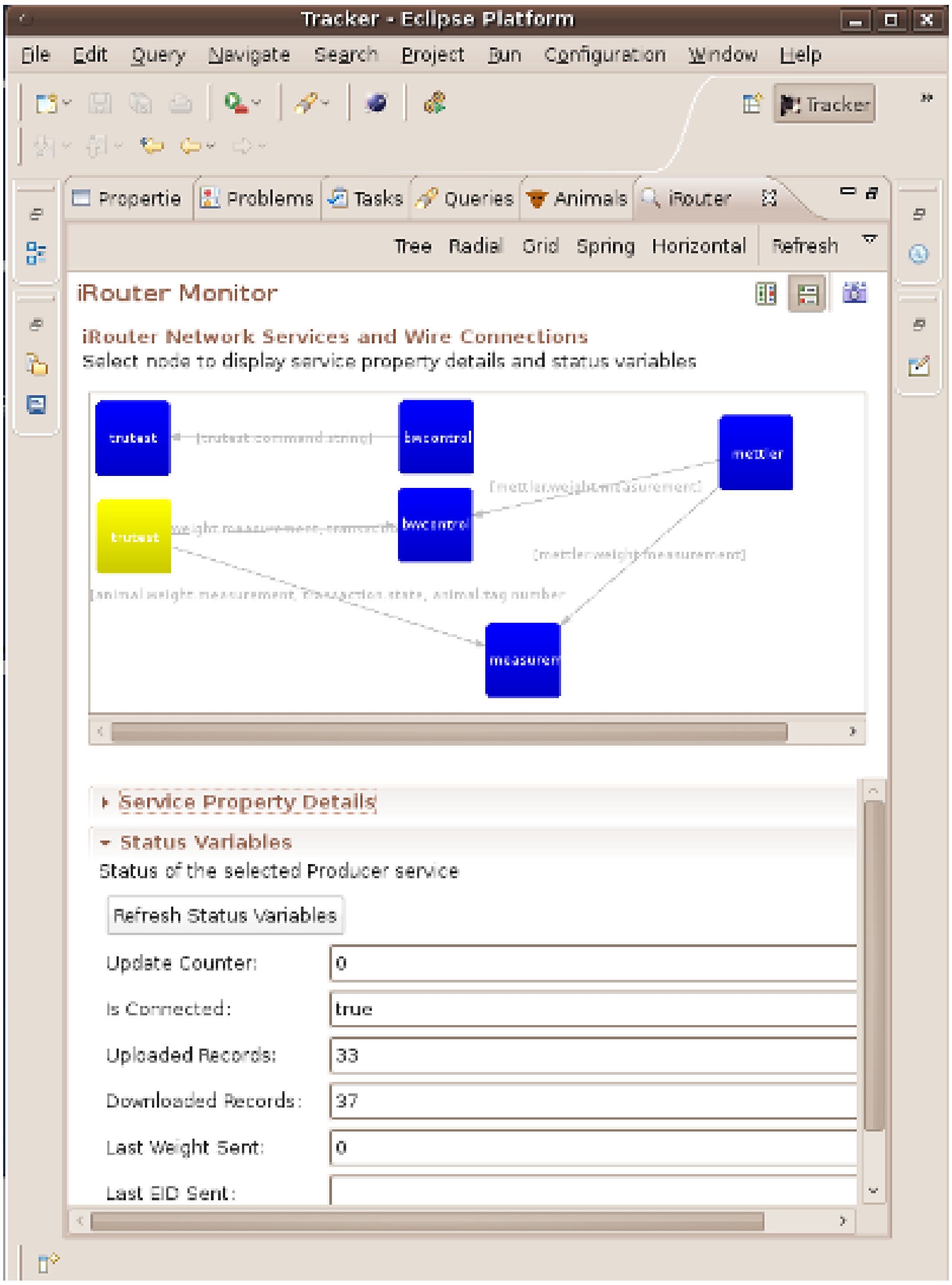
Application Service Monitoring

"To monitor something, one must first see it."

This is especially true when dealing networks of complex iRouter services collaborating within one or multiple *wire groups*. So the main emphasis of the monitoring service is to present:

- A visual representation of the *wire groups*
- Services within each wire group
  - Operation status of a service
- Wire connections between producer and consumer services.

iRouter Monitor view


Tracker - Eclipse Platform
File Edit Query Navigate Search Project Run Configuration Window Help
Tracker
Propertie Problems Tasks Queries Animals iRouter
Tree Radial Grid Spring Horizontal Refresh
trutest
[trutest command string]
bwcontrol
mettler
[mettler.weight.measurement]
trutest
weight measurement, transactio
bwcontrol
[mettler.weight.measurement]
[animal weight measurement, transaction state, animal tag number
measurem

iRouter Monitor

iRouter Network Services and Wire Connections

Select node to display service property details and status variables

Service Property Details

Status Variables

Status of the selected Producer service

Refresh Status Variables

Update Counter:	0
Is Connected:	true
Uploaded Records:	33
Downloaded Records:	37
Last Weight Sent:	0
Last EID Sent:	

Example of the Application monitoring view

The monitor view is a master-detail view with the master page presenting a graph of iRouter service instances. The graph displays the interconnected services as a simplified Functional Flow Block Diagram.

Selecting a node in the graph will show details page of the service containing the sections:

- Service Property Details
  - Showing the static details of the service
- Status Variables
  - Showing the operational status of the service

Wire Group Graph

As configurations are saved or activated *Producer* services are automatically wired to all compatible *Consumer* services. The graphic window of the Monitoring View will dynamically visualize this process with services shown as squares and wires as arrows. To distinguish the type of service Wire arrows are shown pointing from *Producers* to *Consumers*. The visualization is dynamic with services and wires appearing (and disappearing) as they activate and deactivate.

With your mouse you can drag and reposition nodes on the graphic. You can use the view menu to change the sizing of the graphic, the labeled buttons to change the layout style, and the tools in the tool bar to change the orientation of the window. To print the graphic select the camera icon in the toolbar.

Monitoring Status Variables

Selecting a service node in the master tree will display the details of the service in the Status Variables window.

Be aware that iRouter services created from different component factories will have different variables associated with their services. Click on the *Refresh Status Variables* button to see the most current status for the service.

Chapter 7. Component Factories

Component Factories are the creators of the services that do the actual work within the iRouter. There are three types of Component Factories:

1. Business Logic
2. Proxy
3. External

Business Logic Component Factories

Business logic Component Factories create components for transforming, controlling and routing information within the iRouter.

Arithmetic Measurement Converter

Arithmetic Measurement Converter is a generic business logic Component Factory that creates components that convert one measurement to another based on an arithmetic calculation. Arithmetic Measurement Converter:

- Consumes one *Measurement* input
- Produces a *Measurement* product from the consumed *Measurement* and an user configured Arithmetic calculation

Measurement Comparator

Measurement Comparator is a generic business logic Component Factory that creates components that compare two consumed *Measurement* products. Measurement Comparator instances produce a *State* product containing the results of the comparison.

The *Measurement* arriving on the first input is compared with the *Measurement* arriving on the second input. Inputs are configured as *Consumer Scope* parameters. A Measurement Comparator service produces a *State* with a:

- Negative integer value if the first measurement is less than the second
- Zero if the first measurement is equal than the second
- Positive integer if the first measurement is greater than the second measurement.
- Integer.MIN if the states are unequal or incompatible.

Measurement Trigger

Measurement Trigger is a generic business logic Component Factory that creates components that inspect consumed *Measurement* products for conditions. Measurement Trigger instances produce a *State* when specific configured conditions are met. Two types of inspection conditions are supported:

1. Rising Threshold condition will send a trigger *State* when a sequence of consumed *Measurement* products pass through a range of values less than or equal to specified low value up to a value equal to or greater than a specified high value.
2. Falling Threshold condition will send a trigger *State* when a sequence of consumed *Measurement* products pass through a range of values from greater than or equal to specified high value back to a value equal to or less than a specified low value.

The Trigger State will have the following attributes:

- Name
  - com.verticon.tracker.irouter.measurement.trigger
- Value
  - 2

Flow Terminator

Flow Terminator is a generic business logic Component Factory that creates components for controlling the flow of consumed information products passing through the service instance. A Flow Terminator service either passes through or blocks the information passing from the input to the output. Flow control is turned on or off based on a distinct *State* product consumed on a control input.

Flow Terminator has two inputs configured as consumer scopes:

1. Information scope contains information that either is passed through or terminated
2. Control scope that contains control *State* that enable or disable (terminate) the flow of the information.

When pass through is enable the Flow Terminator instance passes through (produces) one product:

1. Identical to the product consumed on its Information scope.

BW Control

BW Control is an livestock *problem domain* and *device* specific custom business logic Component Factory that creates components for controlling animal weight and animal blood weight sensing. BW Control interfaces with a TruTest Scalehead indicator and a Mettler Balance.

The BW Control is included as an example of a custom Component Factory.

Measurement Logger

Measurement logger is a generic business logic Component Factory that creates components for logging Measurements produced by iRouter producer components to the Logback enterprise class logging framework.

Proxy and Gateway Component Factories

Proxy Component Factories create components that interface with external physical:

- Input devices,
- Sensors,
- Displays,
- Actuators.

Gateway Component Factories create services that create information gateways between services running in the iRouter and local services registered in the OSGi framework.

__Proxy and Gateway Component Factories are packaged as optional iRouter features and are documented in the following sections.__

External Component Factories

External Component Factories create instances that are used for documenting external physical devices:

- Input devices,
- Sensors,
- Displays,
- Actuators,
- Gateways

External Component Factories create annotated nodes in the iRouter monitor view. These nodes document the kind, description and location of the external devices that are connected to Proxies and Gateway services.

Note: External Component Factories and the annotated nodes that are created by them are entirely optional. They serve no functional purpose other than documentation.

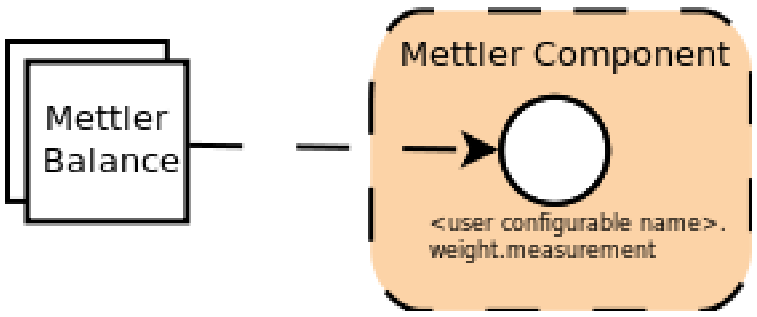
Chapter 8. Mettler Proxy Feature

The Mettler Proxy feature is an iRouter *Proxy Component Factory* that creates user configurable services [11] [19] that interfacing with Mettler Toledo weight balance sensors [12] [19] .

Producer Scopes

Mettler Proxy services produce `wireAdmin` *Measurement* products that are consumed by other iRouter *Business Logic Component* instances. Mettler Service service *Measurement* products contain weight values that represent the value of the item being weighed by the balance. *Measurement* products are identified [13] [19] with a scope id of: `user-configurable-name.weight.measurement` where the prefix `user-configurable-name` can be set by the administrator to distinguish the item being weighed.

Service Diagram



Notes

[11] Mettler Proxy services are implemented as WireAdmin Producer services.

[12] The Tracker Device Simulator product has implemented a simulator of the Mettler balance. This simulator can be used for testing and validating iRouter configurations utilizing Mettler Service Instances.

[13] Scope names identify the information sent by a Producer service to a Consumer service. The iRouter determines connections based on scopes and wire group names.

Chapter 9. TruTest Proxy Feature

The TruTest Proxy feature is an iRouter *Proxy Component Factory* and a *Business Logic Component Factory* for creating services [14] [21] for interfacing with TruTest animal scale-heads.

Proxy Component Factory

The TruTest [15] [21] Proxy Component Factory creates service instances that:

- Synchronize TruTest scalehead animal life data;
  - Animal life data from the TruTest scalehead is sent to a file in a specified user directory
  - Animal life data from the computer is sent to TruTest scalehead as a list of records either from:
    - An iRouter Animal Life Data Component service (For details see the Premises Gateway Feature documentation)
    - A file named animalLifeDataUpLoad.txt in a specified directory on the system file system
- Detect and send animal weights and *Enter* key states
  - TruTest Proxy instances will detect scalehead animal weights and operator keypress information producing:

**_Measurement_ products that contain the weights read by the scalehead

-
  - *State* information indicating that an operator pressed the *Enter* key
- Provide a modular and simple framework for custom TruTest scalehead integration
  - TruTest Proxy instances consume raw TruTest protocol commands produced by other iRouter components for:
    - custom control of the scalehead
    - display of values on the scalehead fields

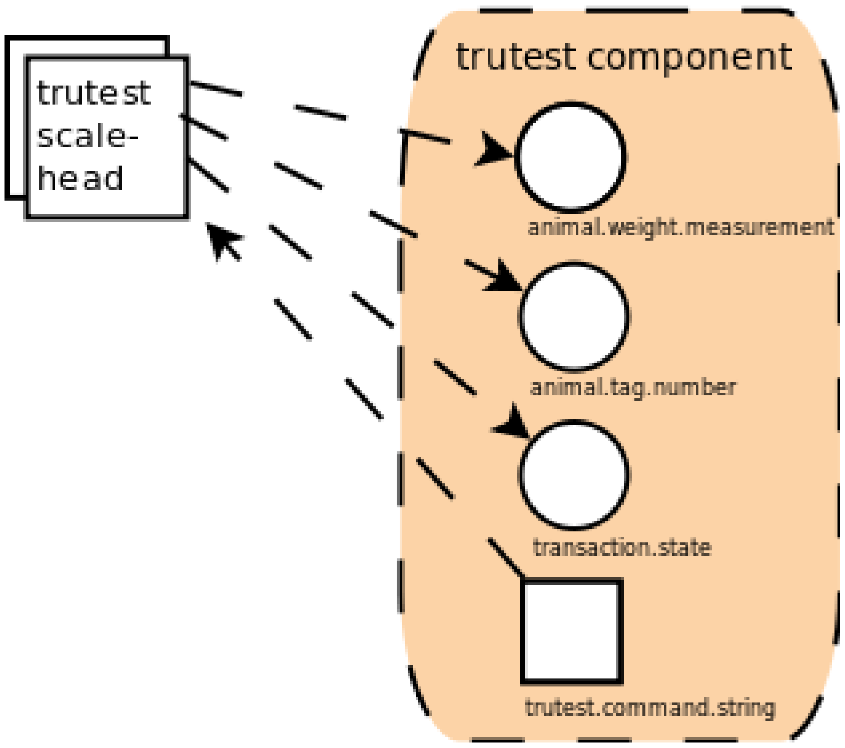
Consumer Scopes

TruTest Proxy Service Instances consume `wireAdmin` products containing TruTest protocol commands which are produced by TruTest Business Logic instances. *See below.*

Producer Scopes

TruTest Service Instances produce `wireAdmin` products that are consumed by iRouter business logic and logging components. TruTest Service Instances produce weight *Measurement* products, animal tag *Numbers*, and key press *State* products.

Service Diagram



Business Logic Component Factories

Measurement Adapter

Measurement Adapter is a *Measurement* Consumer and a TruTest Command producer. It consumes *Measurement* products from generic iRouter component instances and converts these to produce TruTest command products that can be sent to a TruTest Proxy instance for displaying Measurement values in fields on a TruTest Indicator.

EID Adapter

EID Adapter is a Animal ID(Identification) number Consumer and a TruTest Command producer. It consumes generic Animal IDs and converts them into TruTest commands for inputing *EID* values to a TruTest Indicator.

Alarm Adapter

Alarm Adapter consumes *State* products and produces TruTest Command products. It consumes States and converts them into TruTest commands in order to turn on and off the alarm in a TruTest Indicator.

Reception of a distinct *State* sends command products to the TruTest Proxy service instances to turn on the alarm, while reception of any other *State* sends command products to the TruTest Proxy service instance to turn the alarm off.

Notes

[14] TruTest service instances are implemented as WireAdmin Producer and Consumer services.

[15] The Tracker Device Simulator product has implemented a simulator of the TruTest scale head. This simulator can be used for testing and validating iRouter configurations utilizing TruTest Service Instances.

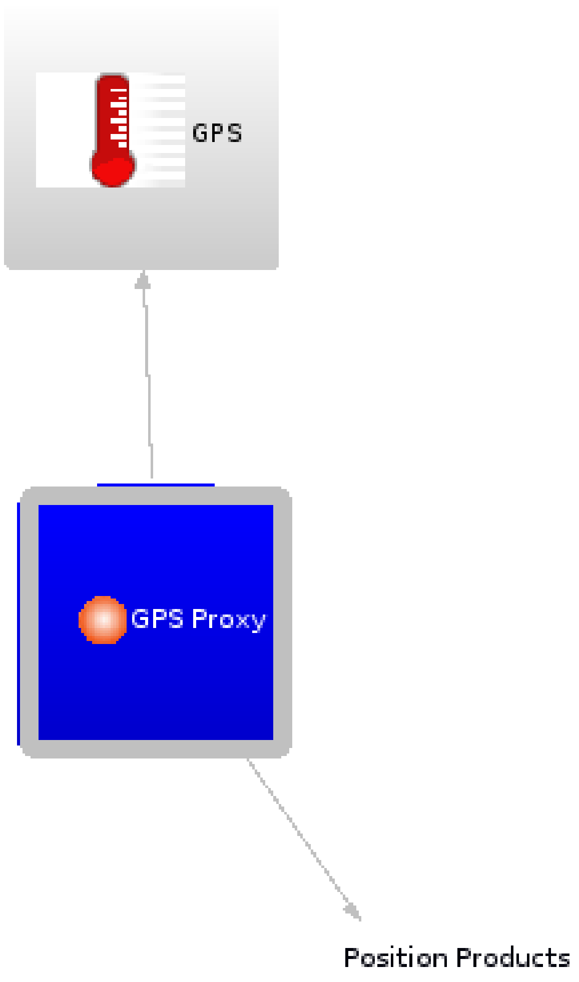
Chapter 10. GPS Proxy Feature

The *GPS* Proxy feature is an iRouter *Proxy Component Factory* that creates user configurable components that interface with GPS devices.

Producer Scopes

GPS Proxy services are implemented as WireAdmin Producer services. 16 [27] A GPS Proxy service connects to a GPS device 17 [27] at a user configurable connection *URI*, parses and validates GPS Fix Data based on the NMEA 0183 $GPGGA sentence and produces an OSGi Position product containing Longitude, Latitude, and Altitude measurements. Position products are consumed by other iRouter *Business Logic Component* instances.

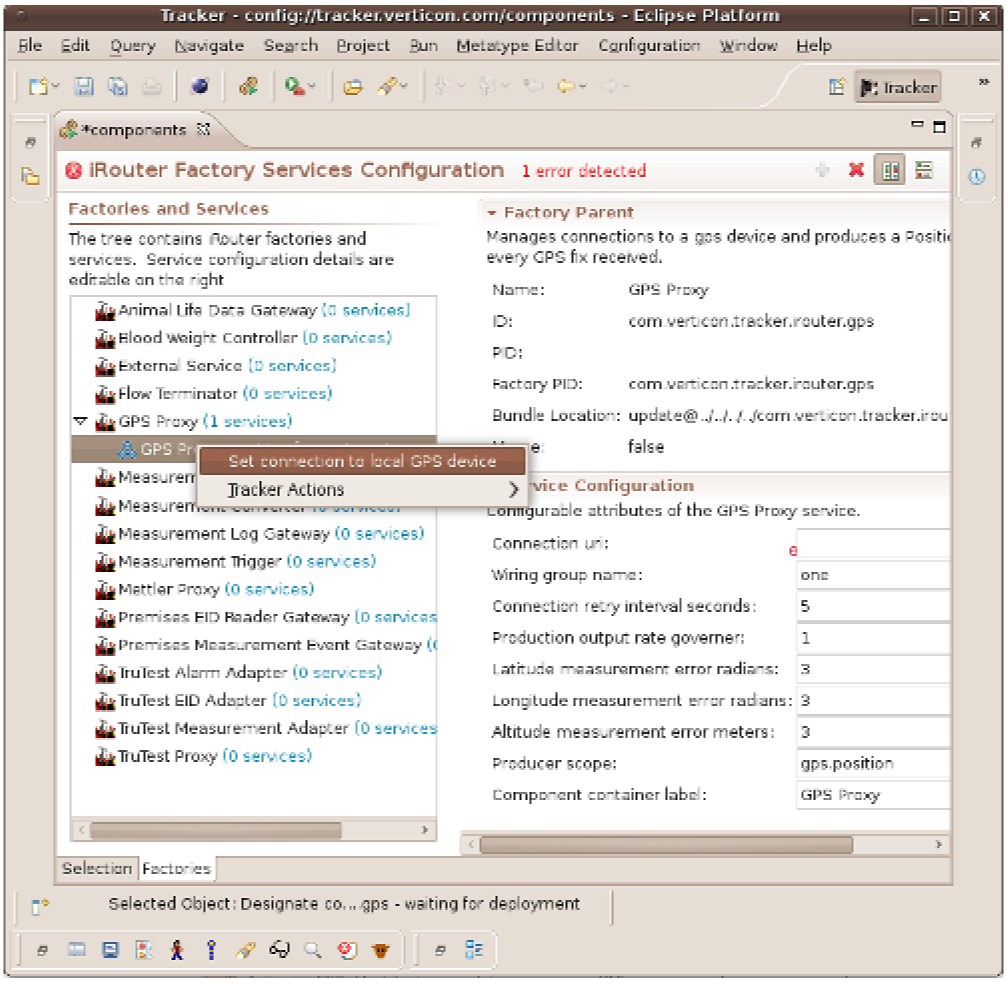
Service Diagram



Creating GPS Components

Like other iRouter components, the Configuration Editor is used to configure and create GPS Proxy component configurations. What makes the GPS component configurations different are the long and archaic looking Connection *URI* attributes that identify the address of a Bluetooth GPS device.

To simplify configuration of Bluetooth GPS device URIs, a *Wizard* for GPS discovery and selection is provided. To use the wizard, first create a new GPS Proxy, select it and right click on it. A pop up menu will be displayed.


Tracker - config://tracker.verticon.com/components - Eclipse Platform
File Edit Query Navigate Search Project Run Metatype Editor Configuration Window Help
Tracker
*components
Set connection to local GPS device
Tracker Actions
Selection
Factories

iRouter Factory Services Configuration 1 error detected

Factories and Services

The tree contains iRouter factories and services. Service configuration details are editable on the right

- Animal Life Data Gateway (0 services)
- Blood Weight Controller (0 services)
- External Service (0 services)
- Flow Terminator (0 services)
- GPS Proxy (1 services)
  - GPS Pr
- Measurem
- Measurem
- Measurement Log Gateway (0 services)
- Measurement Trigger (0 services)
- Mettler Proxy (0 services)
- Premises EID Reader Gateway (0 services
- Premises Measurement Event Gateway (
- TruTest Alarm Adapter (0 services)
- TruTest EID Adapter (0 services)
- TruTest Measurement Adapter (0 services
- TruTest Proxy (0 services)

Factory Parent

Manages connections to a gps device and produces a Positi every GPS fix received.

Name:	GPS Proxy
ID:	com.verticon.tracker.irouter.gps
PID:	
Factory PID:	com.verticon.tracker.irouter.gps
Bundle Location:	update@../../../com.verticon.tracker.irou
e:	false

vice Configuration

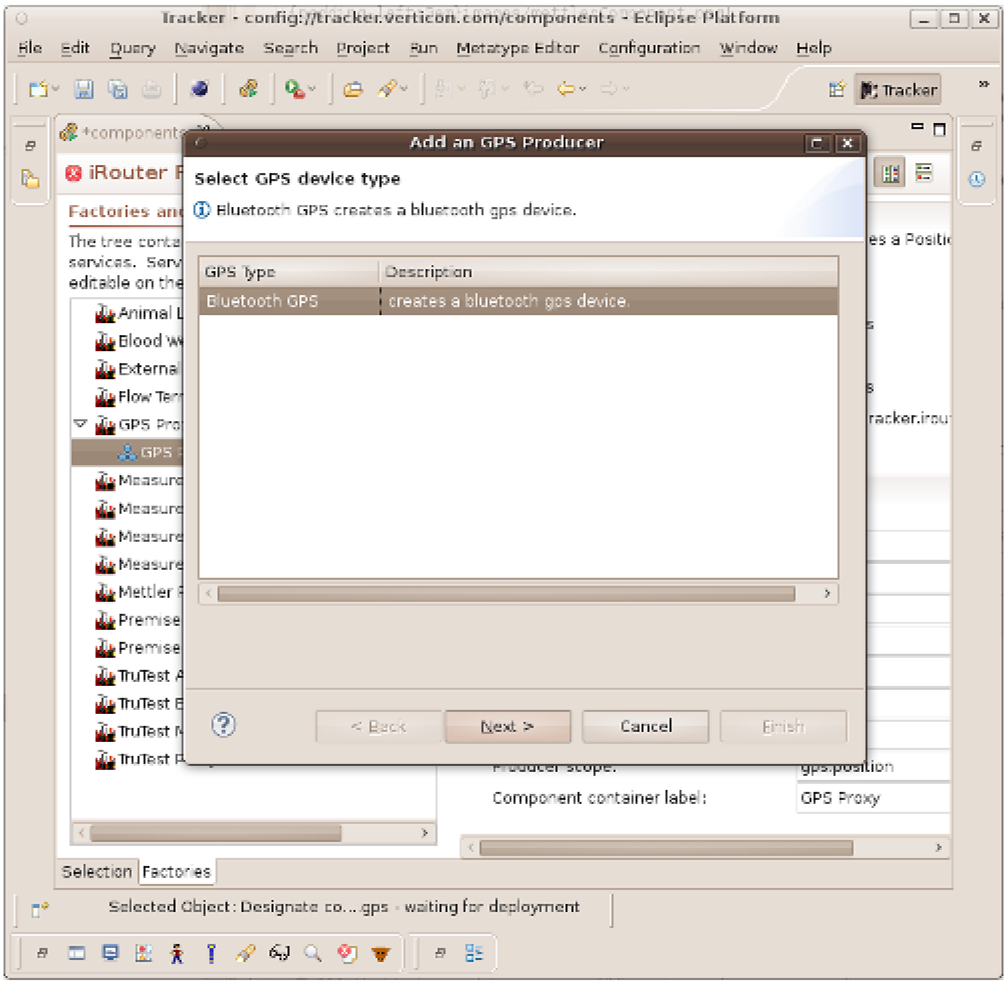
Configurable attributes of the GPS Proxy service.

Connection uri:	
Wiring group name:	one
Connection retry interval seconds:	5
Production output rate governer:	1
Latitude measurement error radians:	3
Longitude measurement error radians:	3
Altitude measurement error meters:	3
Producer scope:	gps.position
Component container label:	GPS Proxy

Selected Object: Designate co....gps - waiting for deployment

Example of the GPS Component PopUp Menu

In the pop up menu, select the `Set connection to local GPS device` item to invoke the Wizard.


Tracker - config://tracker.verticon.com/components - Eclipse Platform
File Edit Query Navigate Search Project Run Metatype Editor Configuration Window Help
Add an GPS Producer
< Back
Next >
Cancel
Finish
Selection
Factories

Select GPS device type

Bluetooth GPS creates a bluetooth gps device.

GPS Type	Description
Bluetooth GPS	creates a bluetooth gps device.

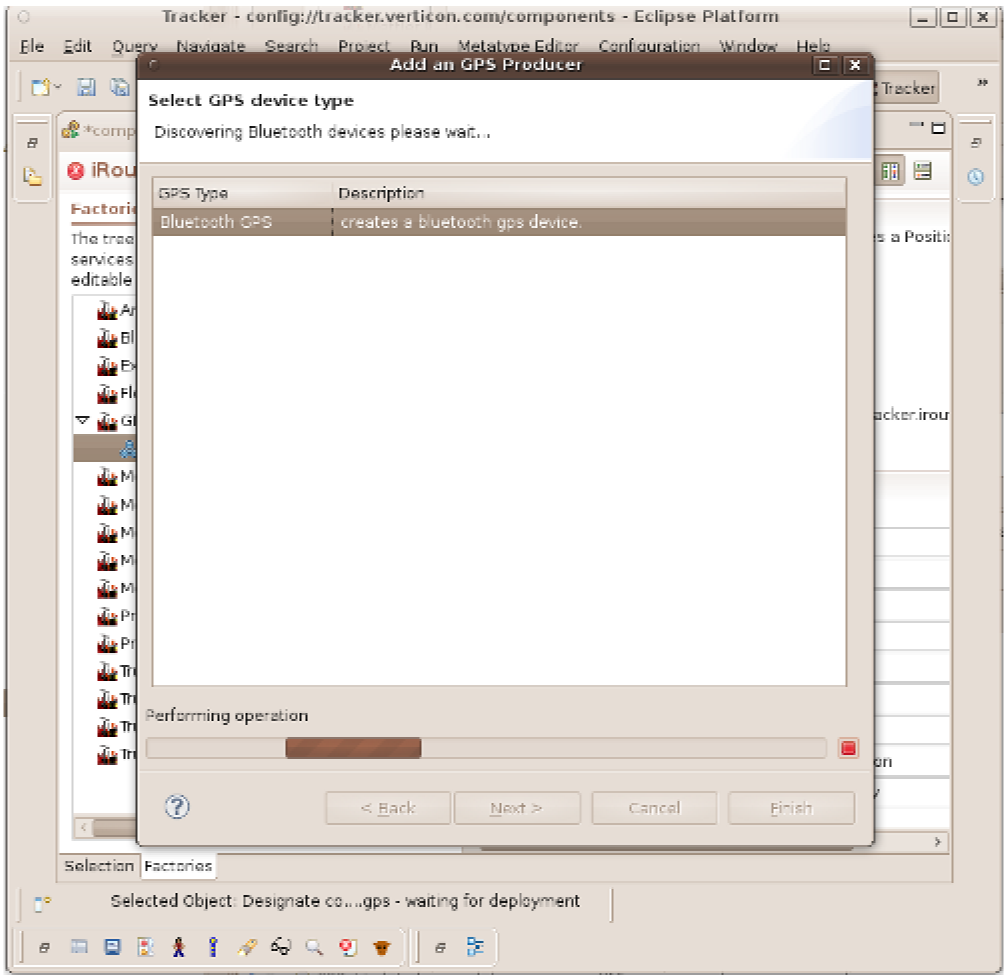
Producer scope: gps.position

Component container label: GPS Proxy

Selected Object: Designate co... gps - waiting for deployment

__Add a GPS Producer Wizard __

To discover all local *Bluetooth* GPS devices, press the `Next` button of the wizard’s dialog. As the wizard pauses as it discovers devices, the progress bar will show activity.


Tracker - config://tracker.verticon.com/components - Eclipse Platform
File Edit Query Navigate Search Project Run Metatype Editor Configuration Window Help
Add an GPS Producer
< Back
Next >
Cancel
Finish
Selection
Factories

Select GPS device type

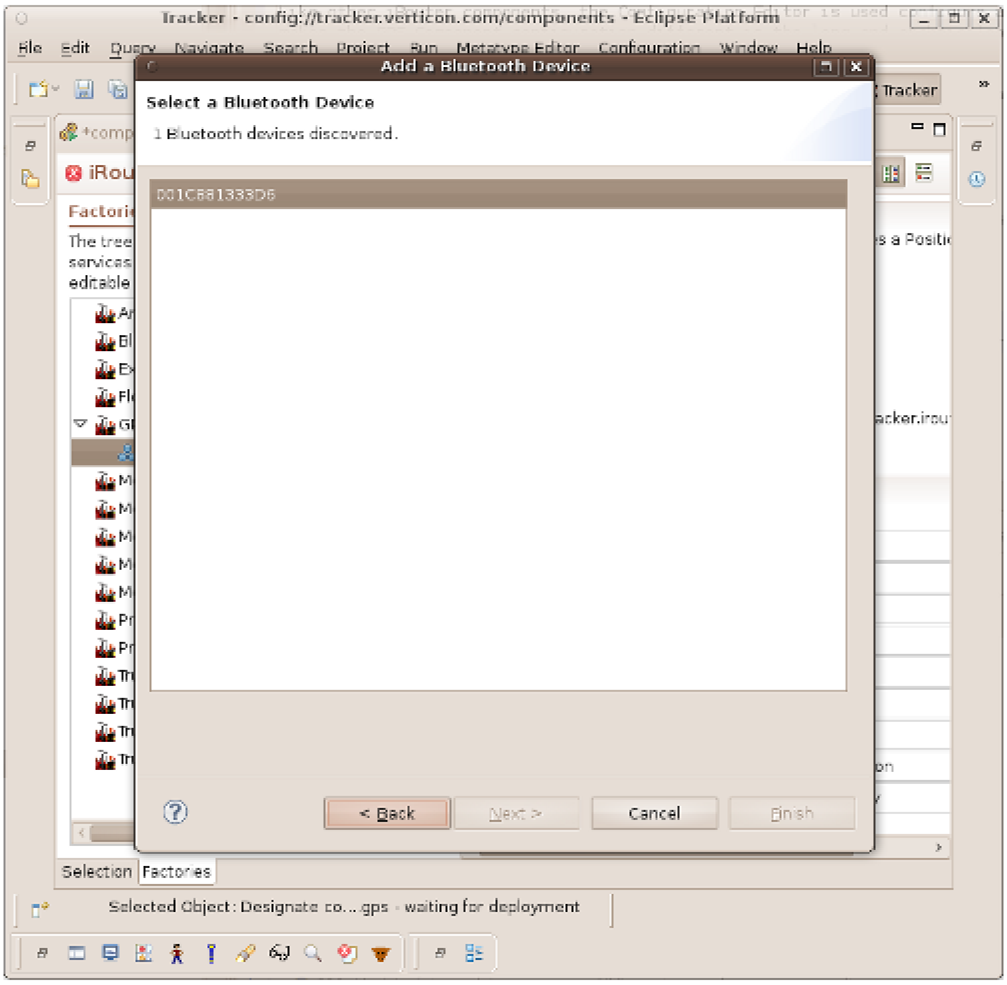
Discovering Bluetooth devices please wait...

GPS Type	Description
Bluetooth GPS	creates a bluetooth gps device.

Performing operation

Selected Object: Designate co....gps - waiting for deployment

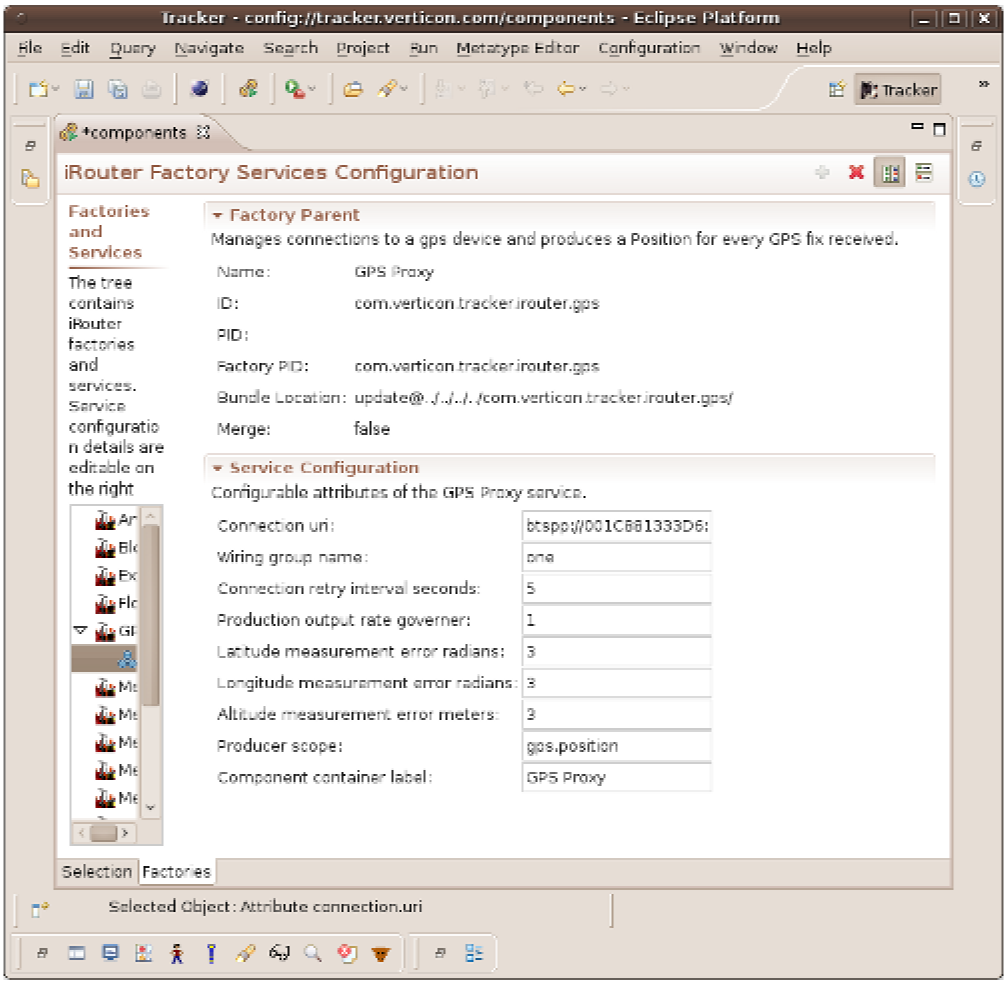
Discovering GPS Devices


Tracker - config://tracker.verticon.com/components - Eclipse Platform
File Edit Query Navigate Search Project Run Metatype Editor Configuration Window Help
Add a Bluetooth Device
Select a Bluetooth Device
1 Bluetooth devices discovered.
001C881333D6
< Back
Next >
Cancel
Finish
Selection
Factories
Selected Object: Designate co....gps - waiting for deployment

Discovered GPS Devices

Select a device from the wizard page's pick list and press the `Next` button. On the next screen, select a service and press finish.

The GPS Connection URI attribute of the GPS Proxy component will be configured with the URI of the device service that was selected.


Tracker - config://tracker.verticon.com/components - Eclipse Platform
File Edit Query Navigate Search Project Run Metatype Editor Configuration Window Help
Tracker
*components
Selection Factories

iRouter Factory Services Configuration

Factories and Services

The tree contains iRouter factories and services. Service configuration details are editable on the right

▾ Factory Parent

Manages connections to a gps device and produces a Position for every GPS fix received.

Name:	GPS Proxy
ID:	com.verticon.tracker.irouter.gps
PID:	
Factory PID:	com.verticon.tracker.irouter.gps
Bundle Location:	update@../../../../com.verticon.tracker.irouter.gps/
Merge:	false

▾ Service Configuration

Configurable attributes of the GPS Proxy service.

Connection uri:	btspp://001C881333D6:
Wiring group name:	one
Connection retry interval seconds:	5
Production output rate governer:	1
Latitude measurement error radians:	3
Longitude measurement error radians:	3
Altitude measurement error meters:	3
Producer scope:	gps.position
Component container label:	GPS Proxy

Selected Object: Attribute connection.uri

Completed GPS Configuration

Notes

[16] Scope names identify the information sent by a Producer service to a Consumer service. The iRouter determines connections based on scopes and wire group names.

[17] The Tracker Device Simulator product has implemented a simulator of the GPS Proxy. The simulator can be used for testing and validating iRouter configurations utilizing GPS Service Instances.

Chapter 11. iRouter Premises Features

There are three iRouter features that facilitate the movement of information between iRouter network services, Premises documents and Premises Editor support tools. The following provides an overview of these features.

For details on these features see the Tracker Business Intelligence User Guide Desktop BI Toolkit Product, Provisioned Features section.

iRouter Premises Gateway

The iRouter Premises Gateway feature is an iRouter feature that facilitates the bidirectional movement of information between *iRouter* services and the *Tracker Premises Documents*.

The Premises Gateway feature contains three iRouter Component Factories and an Gateway EventHandler.

- EID Reader Gateway is a component factory for creating Producer services that transform *EID* numbers read from *RFID* Tag Readers into iRouter *EID number* products.
- Measurement Event Gateway is a component factory for creating gateway services that transform iRouter *Measurement* and *Position* products into EventAdmin Measurement and Position Events.
- Gateway EventHandler a gateway support service that transforms Measurement and Position events coming from the iRouter into Premises animal history events.
- Animal Life Data Gateway is a component factory for synchronizing animal life data to external devices.

iRouter Premises Common Producers

The iRouter Premises Common Producers feature contains two iRouter Component Factories for creating composite Consumer and Producer services that produce iRouter Event and Animal products.

- Premises Event Producer
  - combine measurements, positions, generic events with Animal EIDs and Triggering states to produce iRouter Position and Measurement products that are identified with an animal EID;
- PremisesAnimalProducer
  - combine Animal EIDs, Triggering states and referenced Animal template files to produce iRouter Animal products that can contain one or more of the standard Tracker Premises events.

MongoDB Tracker Store and Location Service

The MongoDB Tracker Store and Location Service feature is a Component Factory that creates an iRouter Consumer service that consumes Tag, Animal and GenericEvent products and records that information to the MongoDB TrackerStore.

Appendix A. Glossary

EID	Electronic animal IDentification
EMF	Eclipse Modeling Facility
GPS	Global Positioning System
RCP	Rich Client Platform
RFID	Radio Frequency ID
URI	Universal Resource Identifier