FIORANO

www.fiorano.com

Fiorano eStudio

User Guide

AMERICA'S
Fiorano Software, Inc.
718 University Avenue Suite
212, Los Gatos,
CA 95032 USA
Tel: +1 408 354 3210
Fax: +1 408 354 0846
Toll-Free: +1 800 663 3621
Email: info@fiorano.com

EMEA
Fiorano Software Ltd.
3000 Hillswood Drive Hillswood
Business Park Chertsey Surrey
KT16 0RS UK
Tel: +44 (0) 1932 895005
Fax: +44 (0) 1932 325413
Email: info_uk@fiorano.com

APAC
Fiorano Software Pte. Ltd.
Level 42, Suntec Tower Three 8
Temasek Boulevard 038988
Singapore
Tel: +65 68292234
Fax: +65 68292235
Email: info_asiapac@fiorano.com

Contents

Chapter 1: Introduction to eStudio

This section outlines some of the key new features added to the Fiorano eStudio:

1. Offline Application Development

With eStudio, you can develop Event Processes without connecting to any server. A server connection is required only when you are deploying an Event Process.

2. EPLCM (Event Process Life Cycle Management)

EPLCM allows an user to move an Event Process in different labeled environments that is, Testing, Staging, QA, and Production at the click of a button. Pre-created profiles for each environment are automatically picked up by the Server at the deployment time.

3. Sub Flows

A powerful new sub-flow concept has been added.

4. Improved Debugger Implementation

Message injection is added, together with a better set of views to simplify debugging.

5. Split File Development for Services and Application

The ServiceDescriptor.xml and Application.xml are changed to split files, which makes them more readable and reduces the memory footprint of eStudio.

6. Service Descriptor Editor

An editor to edit the ServiceDescriptor.xml file. It makes the edit easy as compared a Text/Xml editor.

7. Quicker Custom Property Sheet (CPS) launch

The CPS associated with a given component now launches significantly faster than previous versions of the Studio.

8. Dynamic Validations while Editing and Creating Services and Applications

The Dynamic validations point out errors at development time while Event Processes are being composed, or Services created; errors that had to previously wait until compile or run-time can now be detected earlier in the development/composition cycle.

9. Different Perspective for eMapper and eStudio within Same Workbench

The eStudio incorporates different perspectives for the creation of mappings and for event-process development/editing; user no longer needs to open the Mapper in a separate window and process.

10. UI crafted for Rich User Experience

Significant user feedback has been incorporated within eStudio to provide a rich user-experience. Most common operations can now be performed with a single click, with much less navigation than in previous versions.

11. Support for Version Control Systems

Users can now store applications into any version control system (SVN, CVS, or VSS) using Fiorano eStudio.

12. New Mapping Tool: eMapper

The eStudio incorporates a brand new mapping tool developed ground-up in Eclipse. This new version fixes many bugs over past versions and has several other enhancements.

13. Customization Possible as an Advantage of Eclipse Based Product

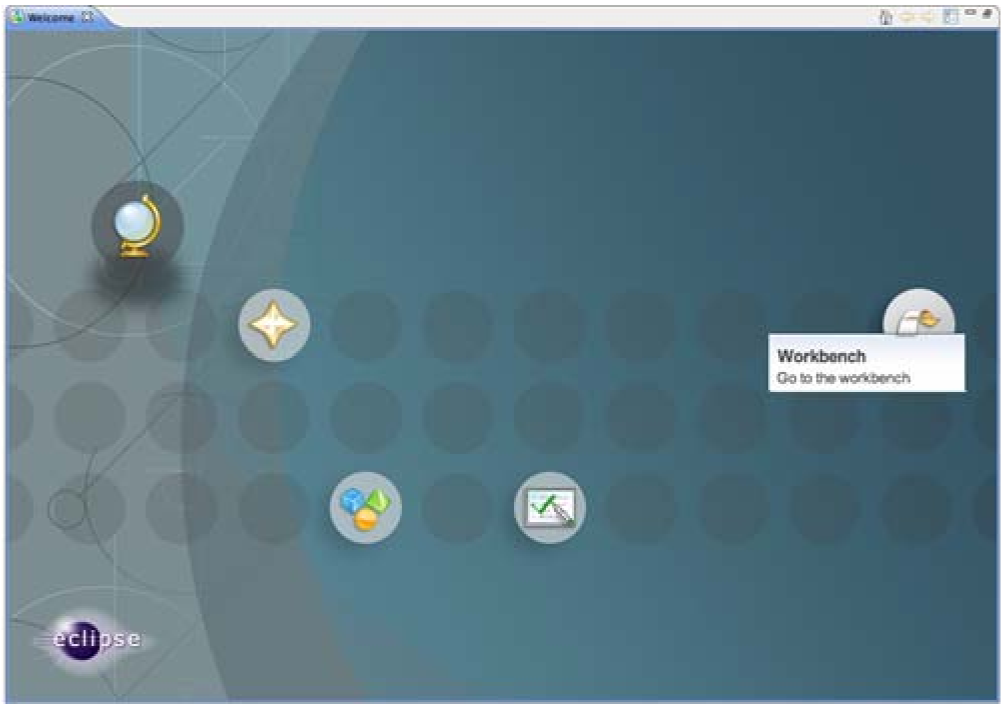
Since eStudio is developed over the Eclipse platform, users can now write their own plug-ins and use existing ones and can also customize the eStudio the way they want. For instance, a user can add a version control plug-in.

1.1 Getting Started with eStudio

Once the Workbench is launched, you see is a **Workspace Launcher** dialog that allows you to select your workspace directory. A workspace is the directory where your work will be stored.

After the workspace location is chosen, a welcome page is displayed. To go to the workbench, click on Workbench icon. A Workbench window offers one or more perspectives. A perspective defines the initial set and layout of views in the Workbench window. Within the window, each perspective shares the same set of editors. Each perspective provides a set of functionality aimed at accomplishing a specific type of task or works with specific types of resources.

For example, the Java perspective combines views that you would commonly use while editing Java source files, while the Debug perspective contains the views that you would use while debugging Java programs. As you work in the Workbench, you will probably switch perspectives frequently.



1.2 Fiorano Perspective

To open Fiorano perspective, perform the following steps:

1. Click the **Open Perspective** button from the shortcut bar on the right-hand side of the Workbench window.
2. Select **Other...** from the drop-down menu.
3. Select the **Fiorano perspective** to open Fiorano perspective. The **Fiorano Navigator**, **Properties**, and **Service Palette** appears.

When the Fiorano perspective opens, the title bar of the window is changed to display the name of the perspective. Additionally, an icon is added to the shortcut bar, allowing you to quickly switch back to that perspective from other perspectives in the same window.

By default, a perspective will open in the same window. We recommend you to open it in a new window by changing the settings in the General > Perspectives preference page.

1.3 Fiorano Navigator

This section explains the Fiorano Navigator views. These views are part of the Fiorano perspective. One important view to become familiar with is one of the navigation views, which displays information about the contents of the Workbench and how the resources relate to each other in a hierarchy.

In the Workbench, all resources are stored in Event Process projects. Projects can contain folders and/or individual files.

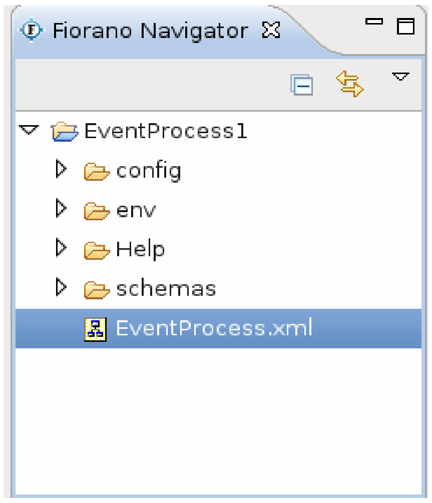


Figure 1.3.1: Fiorano Navigator view

1.4 Fiorano Orchestration

Fiorano perspective in the Workbench is comprised of an editor area Fiorano orchestrator.

For example, when you open a file by double-clicking on **Event Process.xml** in the Fiorano Navigator, it will show the design of the application in Fiorano Orchestrator. You can also open the file in other editors such as, Text Editor, XML Editor, and so on.

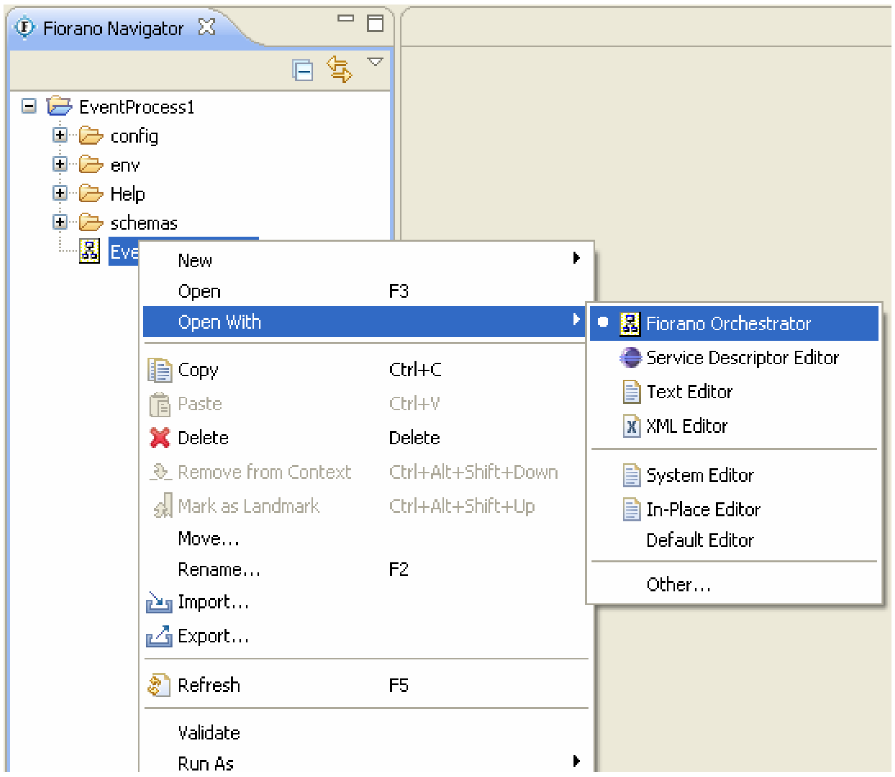


Figure 1.4.1: Fiorano Orchestrator option

1.5 Service Palette

Service Palette shows the services that are present in eStudio repository. The Fiorano Service Palette contains all the Fiorano services such as Bridges, Collaboration, DB, Error, File, and so on.

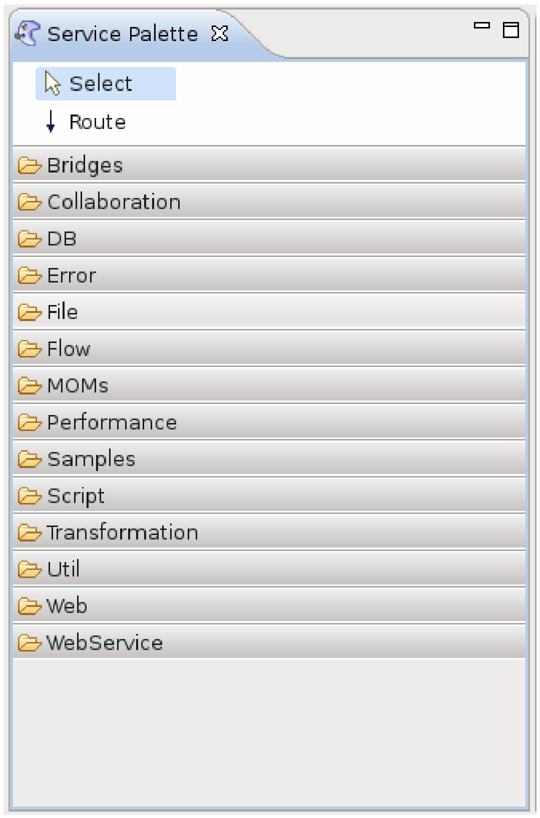
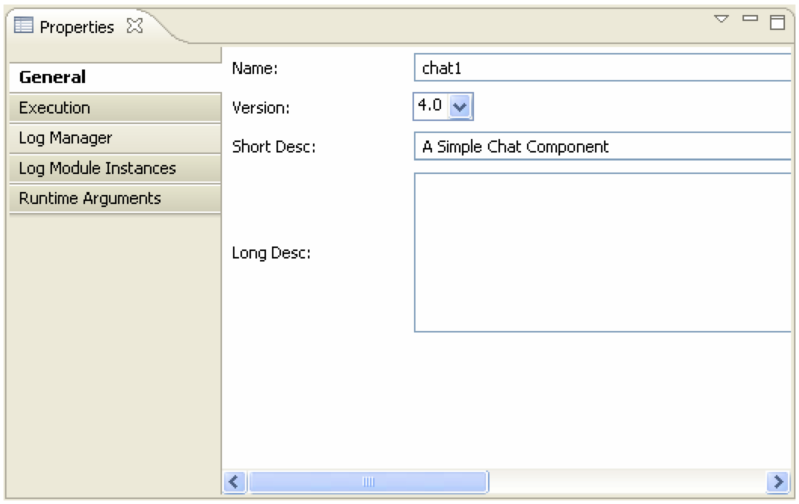


Figure 1.5.1: Service Palette

1.6 Properties

The properties view displays property names and values for a selected item such as, a service instance. To add the Properties view to the current perspective, click **Window** > **Show View** > **Other** > **General** > **Properties**.



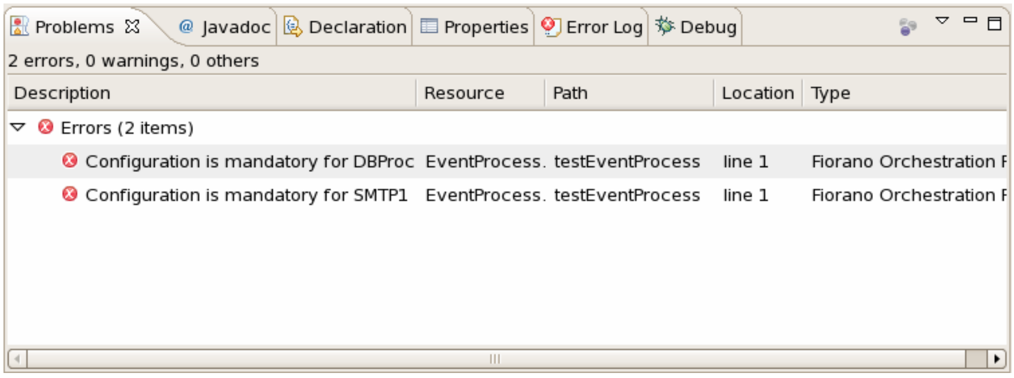
Name: chat1
Version: 4.0
Short Desc: A Simple Chat Component
Long Desc:

Figure 1.6.1: Properties View

1.7 Problems

As you work with Fiorano Processes in the workbench, various builders may automatically log errors or warnings in the **Problems** pane. For example, when you save an Event Process file that contains errors, those errors will be logged in the Problems pane.

To add the Problems pane to the current perspective, click **Window** > **Show View** > **Other** > **General** > **Problems**.



2 errors, 0 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (2 items)				
Configuration is mandatory for DBProc	EventProcess.	testEventProcess	line 1	Fiorano Orchestration F
Configuration is mandatory for SMTP1	EventProcess.	testEventProcess	line 1	Fiorano Orchestration F

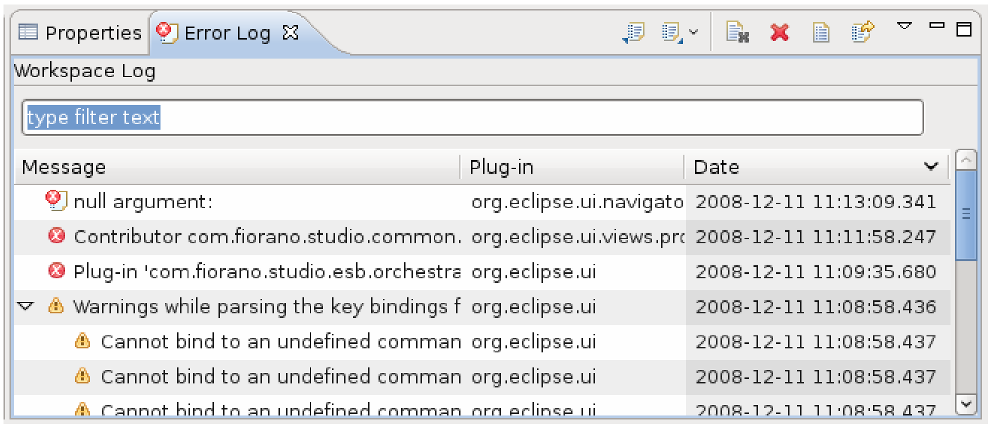
Figure 1.7.1: Problems pane

By default the problems are grouped by severity level. You can also group them by type or grouping can be removed at all. Certain services will add their own grouping. For instance, the Java Development Tools (JDT) support adds a Java Problem Type group. The grouping can be selected using the **Group By** menu.

You can configure the contents of the Problems to view only warnings and errors associated with a particular resource or group of resources. This is done using the Configure Contents dialog available in the drop-down menu. You can add multiple filters to the problems view and enable or disable them as required. Filters can either be additive (any problem that satisfies at least one of the enabled filters will be shown) or exclusive (only problems that satisfy all of the filters will be shown). The two most popular filters (All Errors and Warnings on Selection) are provided by default.

1.8 Error Log

The Error Log view captures all the warnings and errors logged by plug-ins. The underlying log file is a .log file stored in the .metadata subdirectory of the workspace. The Error Log view is available under **Window** > **Show View** > **Error Log**.



Workspace Log

type filter text

Message	Plug-in	Date
null argument:	org.eclipse.ui.navigato	2008-12-11 11:13:09.341
Contributor com.fiorano.studio.common.	org.eclipse.ui.views.prc	2008-12-11 11:11:58.247
Plug-in 'com.fiorano.studio.esb.orchestra	org.eclipse.ui	2008-12-11 11:09:35.680
Warnings while parsing the key bindings f	org.eclipse.ui	2008-12-11 11:08:58.436
Cannot bind to an undefined comman	org.eclipse.ui	2008-12-11 11:08:58.437
Cannot bind to an undefined comman	org.eclipse.ui	2008-12-11 11:08:58.437
Cannot bind to an undefined comman	org.eclipse.ui	2008-12-11 11:08:58.437

Figure 1.8.1: Error log tab

1.9 Fiorano Debugger

Fiorano Debugger view is one of the Fiorano Perspective views. This views will show the list of routes on which debugger is enabled. This gives user to take action on debug message.

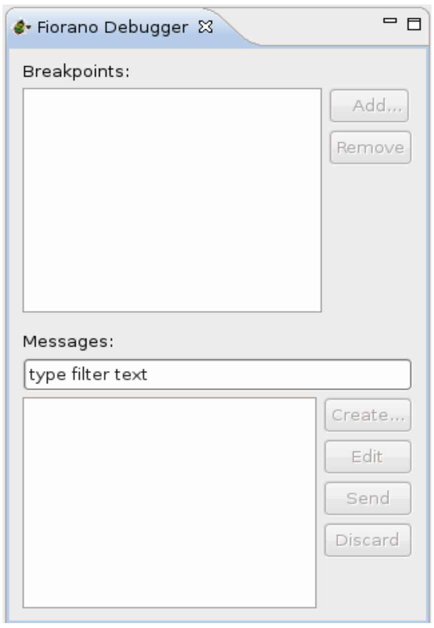
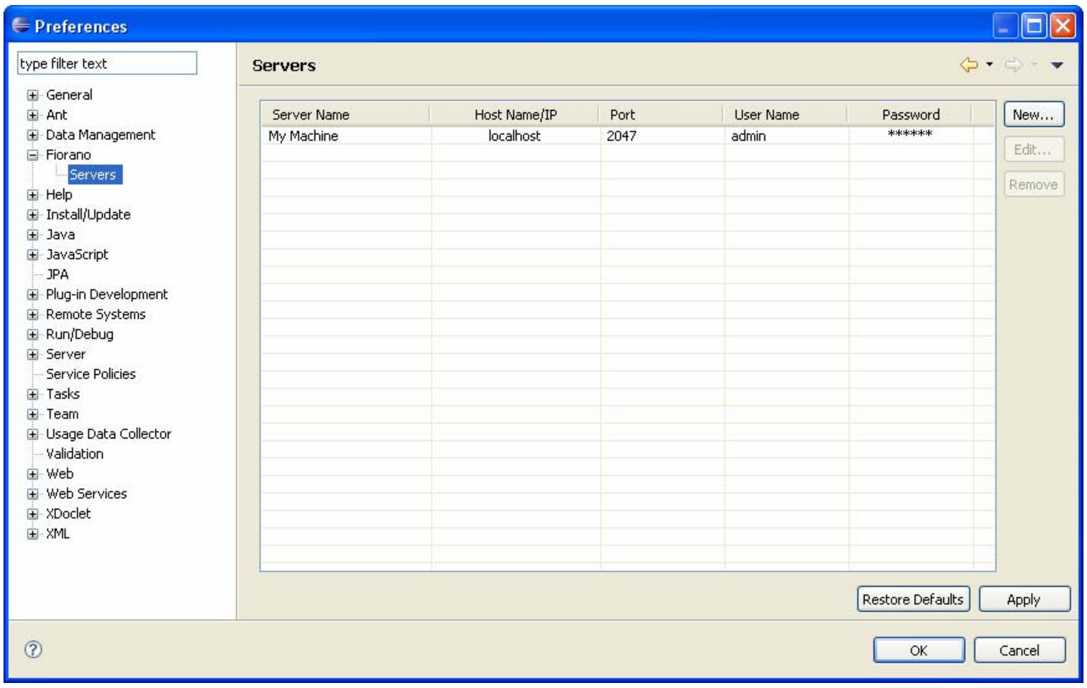


Figure 1.9.1: Fiorano debugger view

1.10 Fiorano Preferences

Fiorano Preferences is available under **Window** > **Preferences** -> **Fiorano**.



Server Name	Host Name/IP	Port	User Name	Password
My Machine	localhost	2047	admin	******

Figure 1.10.1: Preferences dialog box

User can set the server details such as IP of the Enterprise Server and security credentials.

1.11 Service Repository

Fiorano eStudio provides a Service Repository view, which is available under **Window** > **Show View** > **Fiorano** > **Service Repository**. This shows a categorized list of all services. Fiorano eStudio contains its own service repository to enable offline composition of event processes and are not tied to any particular Fiorano ESB server.

Only services which are available in service repository can be used for composing event processes in eStudio. Services can be imported from or exported to file system or a Fiorano ESB server for this view.

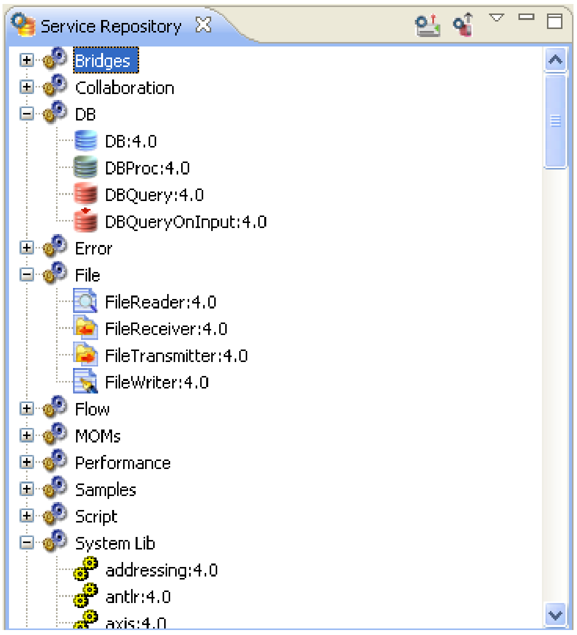


Figure 1.11.1: Service Repository view

1.12 Mapper

The eStudio incorporates Eclipse based Fiorano Mapper as a separate perspective. To open Fiorano Mapper, perform the following steps:

1. Click the **Open Perspective** button from the shortcut bar on the left-hand side of the Workbench window.
2. Select **Other...** from the drop-down menu.
3. Select the **Mapper Perspective** to open Fiorano perspective. The **Mapper perspective** containing **Project Explorer** and **Funclet View** appear.

Chapter 2: Event Processes

2.1 Opening Sample Event Process

To open a pre-configured sample event process, perform the following steps:

1. Navigate to **File** -> **New** -> **Example** and select **Fiorano** -> **Event Process** and click **Next**. A list of pre-configured Event Processes is displayed.
2. Select the Event Process(s), you want to open by selecting the checkbox against each entry and then click **Finish**. Each selected Event Process appears in the **Fiorano Navigator** pane.
3. To see the graphical view of an Event Process, expand the Event Process and double-click the EventProcess.xml file which opens using **Fiorano Orchestrator** plug-in.

2.2 Import and Export Event Processes

You can import and export Event Processes in eStudio. The following sections describe the procedure for Export and Import.

2.2.1 Exporting an Event Process

To export an Event Process, perform the following steps:

1. Right-click the **Event Process** project to be exported and select **Export** from the menu. The Export dialog box appears.

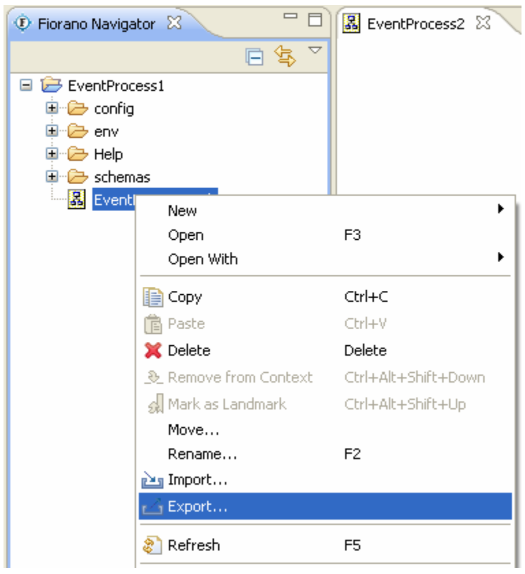


Figure 2.2.1: Export option

2. In the **General** tab, you will see four options, select any one between two options:
   a. **Archive File** if you want to export as an archive file.
   b. **File System** if you want to export as simple directory structure without archiving the exported content.

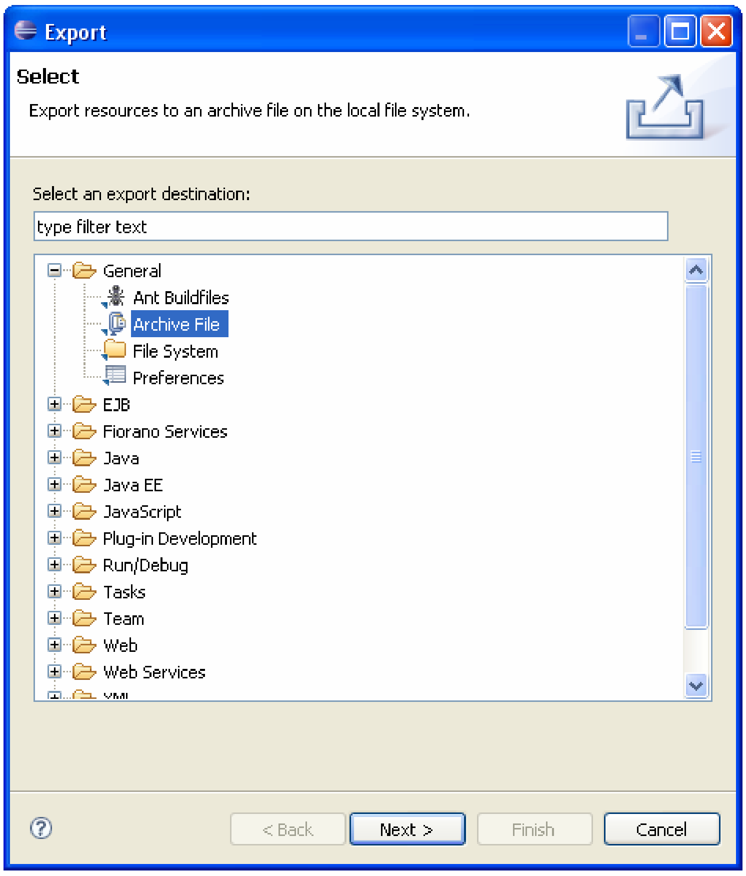


Figure 2.2.2: Export dialog box

3. Select an option and click the **Next** button. Here we have selected **Archive File** option. The **Archive File** dialog box appears.

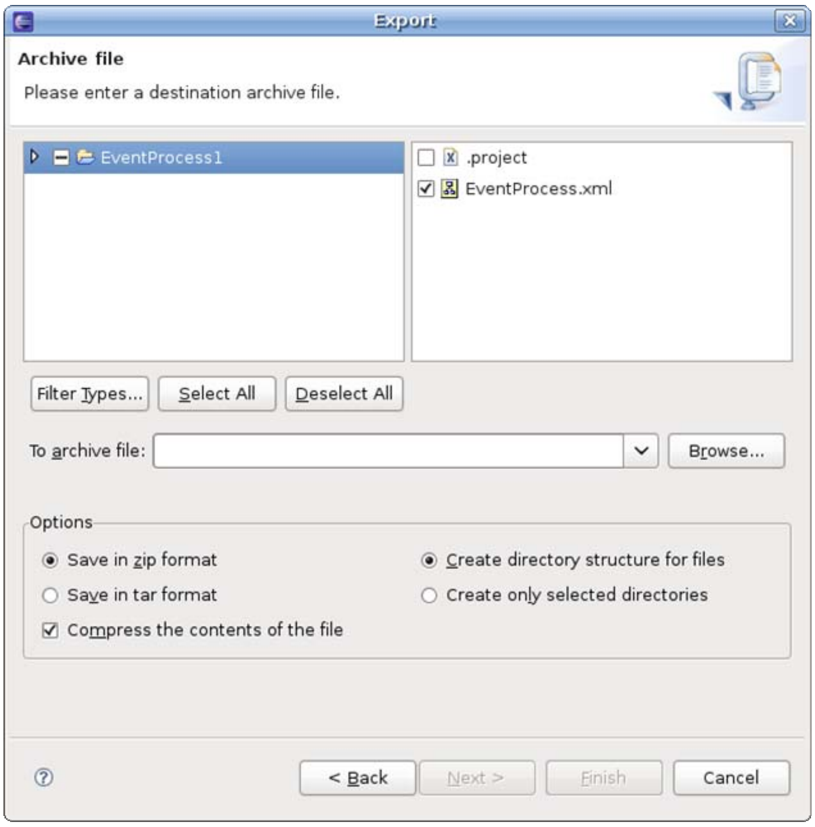


Figure 2.2.3: Archive file destination

4. Modify the selection of projects to be exported. You can select multiple Event Process projects here. Select the project(s), you wish to export and then specify the target file/directory name by clicking the **Browse** button.

5. Click the **Finish** button. The Event Process(s) is now saved at the target location.

2.2.2 Importing an Event Process

To import an Event Process created in eStudio, perform the following steps:

1. From the **File** menu, click the **Import** option The **Import** dialog box appears.
2. From the **General** tab, select **Existing Project into Workspace** option and click the **Next** button. The Import Projects panel appears.
3. Here you will see two options:
   a. **Select root directory** - Select this option if the Event Process(s) to be imported exists as a directory in the file system.
   b. **Select archive file** - Select this option if the Event Process(s) to be imported exists as an archive file in the file system.
4. After choosing one of the above options, a list of projects will appear. Select the projects, you wish to import and then click the **Finish** button. The selected projects will be imported into the workspace.

Note: The projects which already exist in the workspace will not be shown when trying to import them.

2.3 Creating New Event Process

To create a new Event Process, perform the following steps:

1. From the **File** menu and select **New** -> **Event Process Project**. Specify the name of the Event Process project and click **Finish**. The specified Event Process will appear as **Event Process GUID** and display name. The new project appears in the **Fiorano Navigator** pane.
2. Expand the Event Process and double-click on **EventProcess.xml** file to open the file in Fiorano Orchestration Editor. For information on composing an Event Process, see Chapter 3: Composing an Event Process.

Chapter 3: Composing Event Processes

Composition of Event Processes is based on the component-based programming model. An Event Process is composed of services (also known as Business Components) linked to each other by Data Routes.

Event processes are designed by **drag-drop-connect** of service components. The components are customized by configuration rather than custom code. The routes between components are drawn by visually connecting the component ports. Every component instance in the flow can be configured to be deployed on different nodes of the ESB network.

The following sections describe how to compose an Event Process, adding remote service instances, adding external Event Processes. The sample Event Process illustrated below connects Fiorano Chat Business Components with bidirectional Event Routes. The two instances will be configured to run on different nodes in the network.

3.1 Adding Components

To add components, perform the following steps:

1. Open the Service Palette and click the Category tab (say **Collaboration**) corresponding to the service.
2. Drag and drop the business component icon (**Chat**) onto **the Event Process** editor.
3. Each icon in the Event Process editor represents an instance of the Business Component. By default, the name of each instance of the Business Component is the Business Component GUID followed by the instance ID count. You can rename the Business Component instance, when required.

3.2 Connecting to Routes

For data to flow between two Business Component instances, they need to be linked through Event Routes. The Route represents the Brokered Peer to Peer data Route.

Event Routes are unidirectional and always originate from output Event Port of the source Business Component and end at input Event Port of the target Business Component.

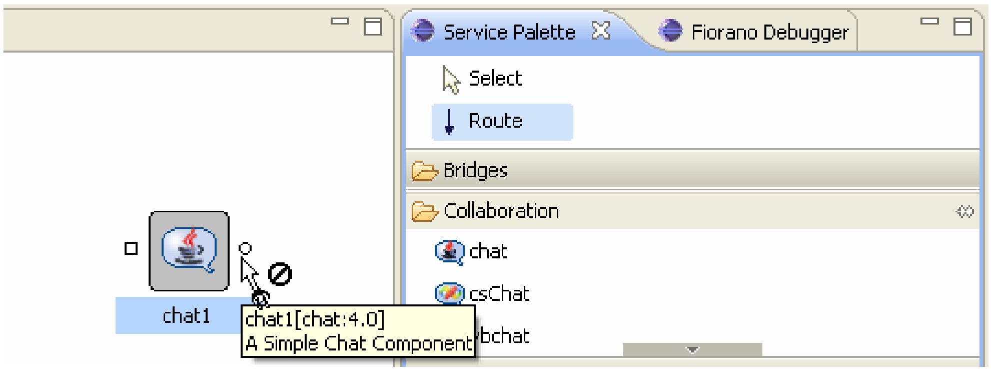


Figure 3.2.1: Conecting to Routes

Connect the Route from the output channel (OUT_PORT) of Chat1 Business Component icon to the input channel (IN_PORT) of the Chat2 Business Component icon and vice versa, as shown in the Figure 3.2.2. By default, each Route is identified by an Event Route name such as, Route1 and Route2. The suffix represents the instance count of the Route. You can edit the Route name using the Properties window.

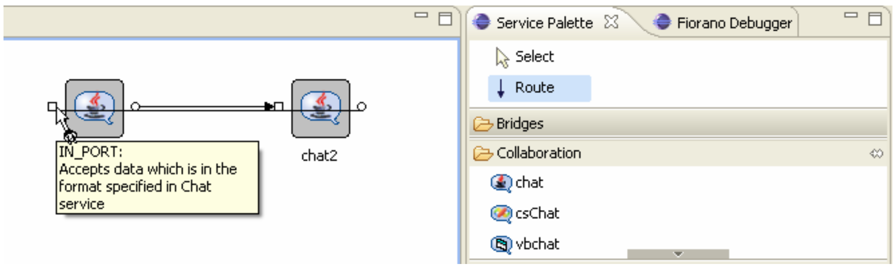


Figure 3.2.2: Business Component instances

3.3 Adding Remote Service Instance

The Fiorano SOA Platform allows the user to compose an Event Process with Business Component instances from other Event Processes. This enables the Fiorano Event Process Orchestrator from reusing a single Business Component instance across Event Processes to reduce Business Component instance redundancy.

To add a remote service instance, perform the following steps:

1. Click the **Insert Element** into **Event Process** icon and select the option **Insert Remote Service Instance** from the drop-down list.

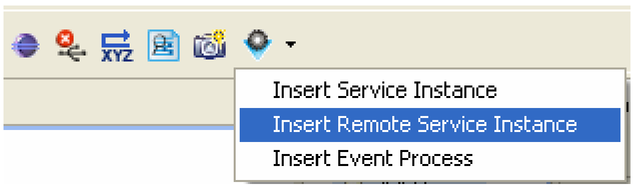
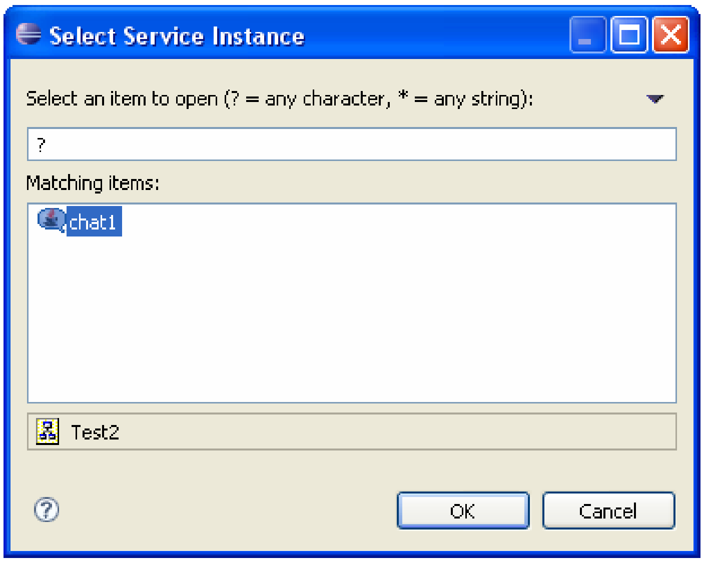


Figure 3.3.1: Insert Remote Service Instance option

2. The External Business Component Selection wizard starts, as shown in Figure 3.3.2.


Select Service Instance

Select an item to open (? = any character, * = any string):

?

Matching items:

chat1

Test2

Figure 3.3.2: Select Service Instance

3. Select the service instance, you want to add as Remote Service Instance and click the **OK** button.
4. The Remote service is added to your Event Process with a satellite like icon in the component as shown in the Figure 3.3.3.

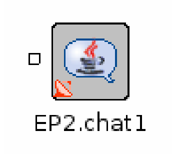


Figure 3.3.3: Remote service

You can use the External Business Component instance similar to normal Business Component instance. You can create routes between any of the Event Process Business Component instances and the input ports of the External Business Component.

3.4 Adding External Event Process

To add an External Event Process, perform the following:

1. Click the **Insert Element** into **Event Process** icon and select the option **Insert Event Process** from the drop-down list. The **Select Event Process** dialog box appears.

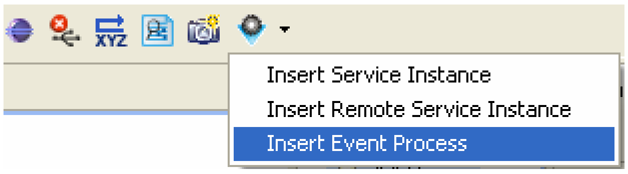
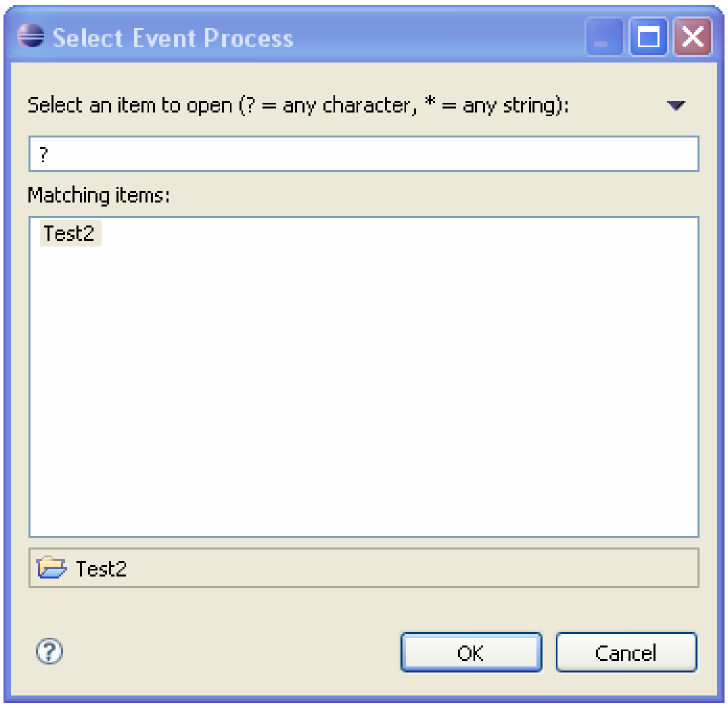


Figure 3.4.1: Insert Event Process

2. Select the Event Process from the list and click the **OK** button.


Select Event Process

Select an item to open (? = any character, * = any string):

?

Matching items:

Test2

Test2

Figure 3.4.2: Select Event Process dialog box

3. The Event process instance representation appears on the Event Process editor, as shown in Figure 3.4.3.

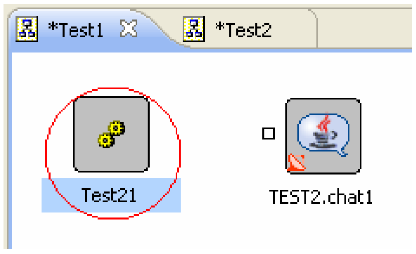
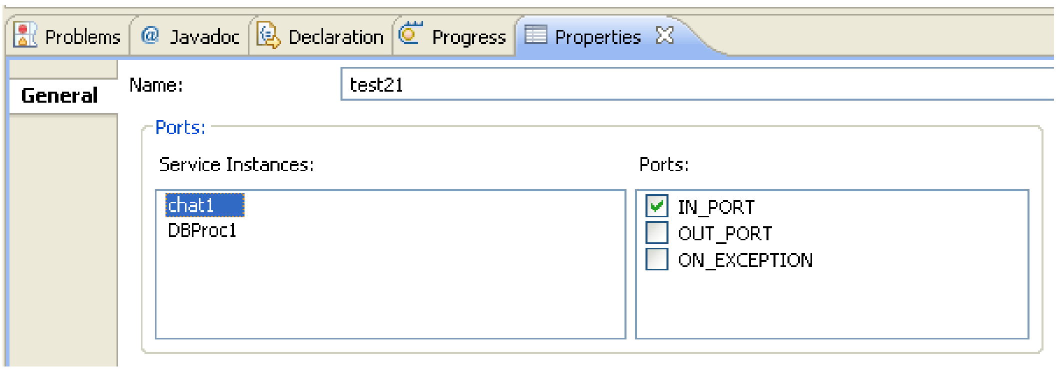


Figure 3.4.3: Event Process editor

B default no input and output ports are shown for an Event Process instance. The user can add the required input and output ports to the Event Process instance from the Properties tab as shown in Figure 3.4.4.



Name: test21

Ports:

Service Instances:
chat1
DBProc1

Ports:
- [✓] IN_PORT
- [ ] OUT_PORT
- [ ] ON_EXCEPTION

Figure 3.4.4: Properties tab

3.5 Configuring Document Tracking

The eStudio provides you with a state-based workflow view that enables tracking and monitoring of documents from one state to another. Additionally, you can define the state an Event Process should be enabled or disabled for tracking.

3.5.1 Defining Document Tracking

A workflow is defined as the Event Process scope within which a large number of documents and messages flow. Whenever a new document enters the workflow, a new workflow instance is generated. Each workflow instance has a unique ID, assigned by the FMP environment. In a state enabled workflow, all the states that these workflow instances traverse are stored for tracking purpose.

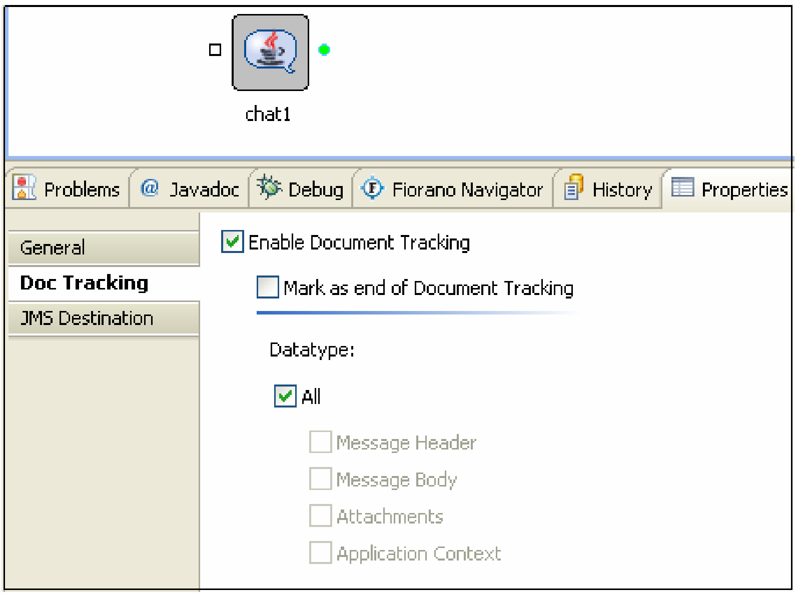
Each workflow instance contains information about documents that flow through a particular workflow instance. Each time a document passes through a track-able state, a state event is generated and the document is given a new Document ID by that track-able state. You can view all the information related to a document, using Fiorano Web Console.

3.5.2 Enabling State Based Tracking

To enable tracking for a particular state:

1. Select the Event Port, you want to configure. The **Properties** pane appears (if the Properties pane does not appear, go to Window->Show View-> Others-> and select Properties).

2. Enable the **Enable Document Tracking** option in **Doc Tracking** tab.



☑ Enable Document Tracking

☐ Mark as end of Document Tracking

Datatype:

☑ All

☐ Message Header

☐ Message Body

☐ Attachments

☐ Application Context

Figure 3.5.1: Doc Tracking tab

3. Ports with enabled document tracking are filled with green color, as shown in the Figure 3.5.2. This indicates that state tracking for that particular Event Port is enabled. All messages passing in and out of that Event Port will be tracked.

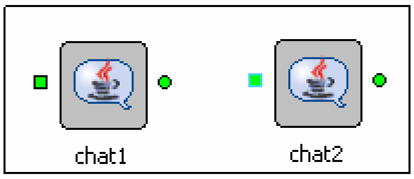


Figure 3.5.2: Ports colors

In the Chat Event Process, as shown in the Figure 3.5.2, the state tracking for the following states is enabled:

IN_PORT of chat1

OUT_PORT of chat1

IN_PORT of chat2

OUT_PORT of chat2

3.5.3 Setting the End State

To track a workflow, you need to mark an end state for the workflow. The end state, as the name implies, marks the end of the workflow.

To enable document tracking using the button provided in toolbar, perform the following steps:

1. Click the **Toggles the Document Tracking Mode** button from the toolbar and click the ports of documents to be tracked.

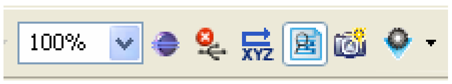
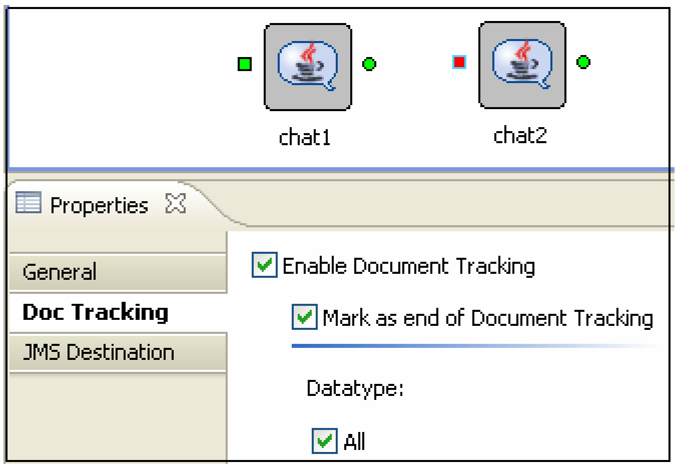


Figure 3.5.3: Toggles the Document Tracking Mode button

2. Click the Event Port to be configured. In the Doc Tracking tab, enable the **Mark as end of Document Tracking** option. The Last port which has been selected will be filled with red color, meaning end of document tracking.



- [✓] Enable Document Tracking
  - [✓] Mark as end of Document Tracking

Datatype:

- [✓] All

Figure 3.5.4: Mark as end of Document Tracking option

3.6 Configuring Selectors on Routes

The eStudio allows you to define Selectors for the flow of data through an event route. Take the example of an Event Process containing two instances of a Chat business component and an instance of a Display business component, as shown in Figure 3.6.1.

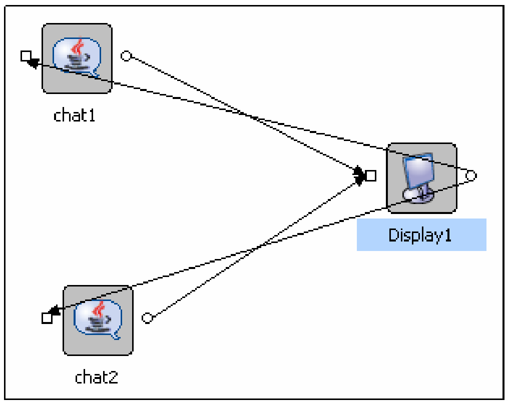


Figure 3.6.1: Out-port of chat1 and chat 2 to display in-port, out-port of display to in-port of Chat1 and chat2

In the above event process, the event routes exist as defined below:

- **Route1:** Connects OUT_PORT of Display1 to IN_PORT of Chat2
- **Route2:** Connects OUT_PORT of Display1 to IN_PORT of Chat1
- **Route3:** Connects OUT_PORT of Chat1 to IN_PORT of Display1
- **Route4:** Connects OUT_PORT of Chat2 to IN_PORT of Display1

Now, let us define conditional data flow from the Display business component instance. Assume that Display1 sends only those messages on Route1 which are meant for Chat2. In other words, it forwards only those messages on this Route, which have originated from Chat2. Similar conditions should also apply to Chat1. It should also receive only those messages that it sends to Display1.

To define conditional flow of data through route1, perform the following steps:

1. Select Route 1, the properties of this route is displayed in the **Properties** tab.
2. Click **Selector** option and choose the option **chat2** from **Sender** properties as show in Figure 3.6.2.

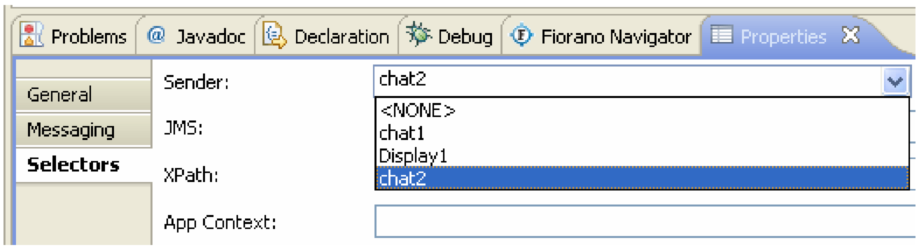


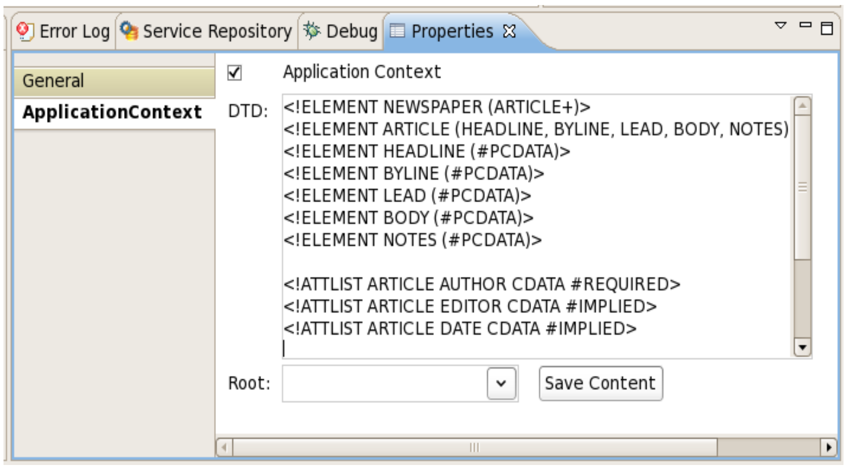
Figure 3.6.2: Selectors option

3. This ensures that data intended for only Chat2 will travel through this route1. Similarly, set this value to Chat1 for Route2. This ascertains conditional flow of data.

3.7 Configuring Application Context

Defining Application Context for an application:

1. In the Event Process Project, click the **Orchestration Editor** and open the properties view.
2. Form the **Properties** tab, click the **ApplicationContext** tab of the view.
3. Enable the **Application Context** option.
4. Provide DTD content in the DTD text section.
5. Click the **Save Content** Button to save the changes.
6. Select root element form the list of available roots in the root Combo.



```
<!ELEMENT NEWSPAPER (ARTICLE+)>
<!ELEMENT ARTICLE (HEADLINE, BYLINE, LEAD, BODY, NOTES)
<!ELEMENT HEADLINE (#PCDATA)>
<!ELEMENT BYLINE (#PCDATA)>
<!ELEMENT LEAD (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
<!ELEMENT NOTES (#PCDATA)>

<!ATTLIST ARTICLE AUTHOR CDATA #REQUIRED>
<!ATTLIST ARTICLE EDITOR CDATA #IMPLIED>
<!ATTLIST ARTICLE DATE CDATA #IMPLIED>
```

Figure 3.7.1: ApllicationContext option

Chapter 4: Event Process Life Cycle Management

The Event Process Life Cycle Management refers to deployment of an Event Process in various environments like Development, Testing, Staging, and Production. The user does not have to create different Event Processes for different environments; instead the user can simply specify the properties for service instances comprising Event Process for various environments in a single Event Process.

4.1 Using Event Process Life Cycle Management

4.1.1 Setting Properties of Service Instances for Different Environments

Select the **CPS Target Environment** in the properties of the Event Process comprising the service instance. Hereafter, environment dependent service properties will be written to the corresponding env.xml file and will be picked up from that file when Event Process is launched in that particular environment. You can specify these properties for more than one environment by switching the CPS Target Environment label in properties of an event process. This way service instances can have different set of properties while running on different environments.

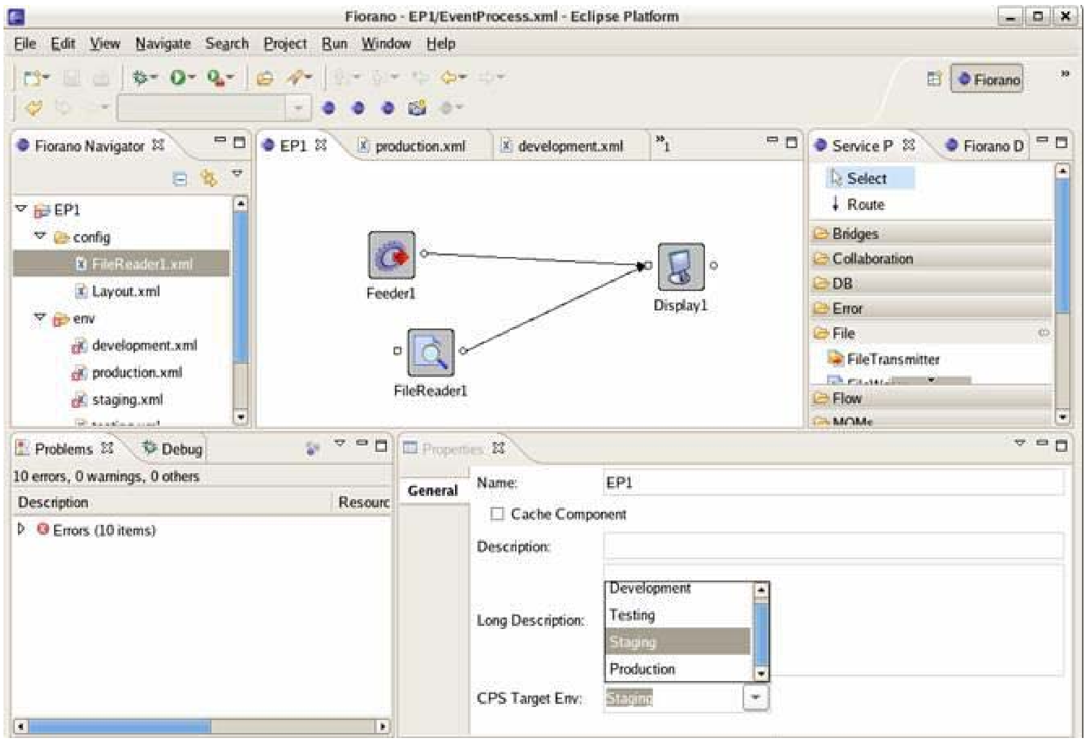
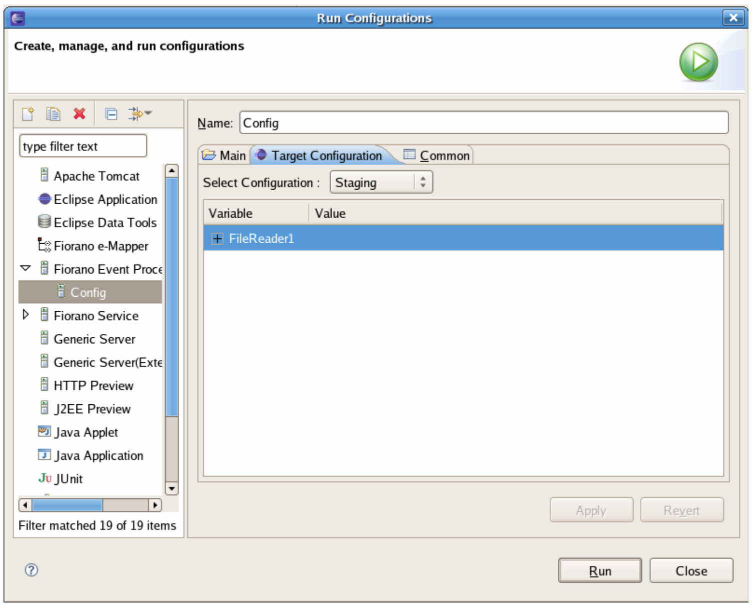

Name: EP1
☐ Cache Component
Description:
Long Description:
CPS Target Env: Staging

Figure 4.1.1: Eclipse Platform

4.1.2 Running an Event Process on an Environment

To run an Event Process on a particular environment, perform the following steps:

1. Right-click the Event Process window and select **Run Configurations**. The Run Configurations appears, as shown in Figure 7.1.2.



Run Configurations

Create, manage, and run configurations

Name: Config

Select Configuration : Staging

Variable	Value
FileReader1	

Filter matched 19 of 19 items

Figure 4.1.2: Run Configurations option

2. Add a new configuration to run Event Process by double-clicking **Fiorano Event Process** option.

3. From the **Main** tab, select the Event Process to be launched and select the server on which the event process has to be launched.

4. Now, from the **Target Configuration** tab, select the environment in which the Event Process should be launched.

5. On selecting the environment label for Event Process, you can see the properties of service instances of an Event Process for the corresponding environment.

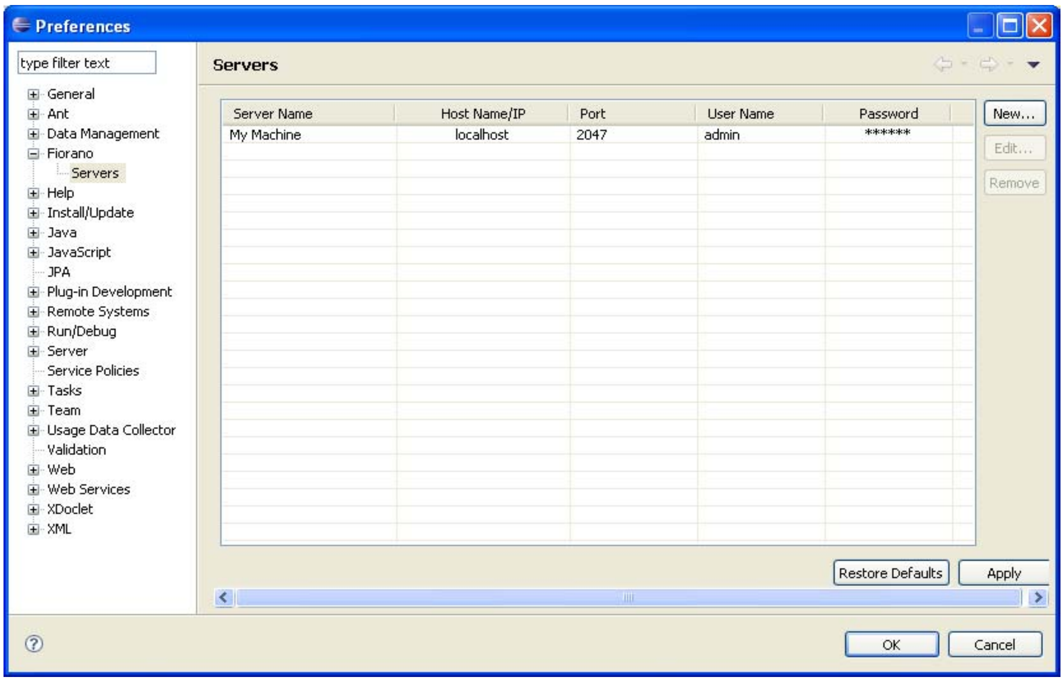
Note: These properties cannot be edited; they can only be edited from the **Properties** view **Deployment** tab.

Chapter 5: Running Event Process

The following sections describe the process to run an Event Process. Before launching the Event Process, Fiorano Enterprise Server and Fiorano Peer Server should be running.

5.1 Defining Servers

1. To define a new server, navigate to **Window – > Preferences → Fiorano → Servers**.



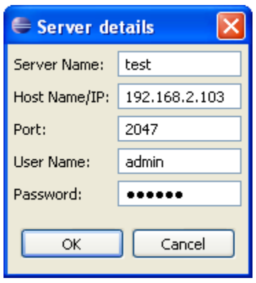
Servers

Server Name	Host Name/IP	Port	User Name	Password
My Machine	localhost	2047	admin	******

Figure 5.1.1: Preference dialog box

2. By default, the Enterprise server running on the local machine is added with default values for Port, User name, Password. If the default values are changed, use the **Edit** button to edit these values.

3. Click the **New** button to add a new Enterprise server. The **Server Details** dialog box appears.



Server Name: test
Host Name/IP: 192.168.2.103
Port: 2047
User Name: admin
Password: ••••••

Figure 5.1.2: Server Details dialog box

4. Provide the remote Enterprise server details and click **OK**. Then click **Apply** and **OK** to save the new server configuration.

5.2 Configuring Nodes for Services

The service can be launched on any of the available peer servers connected to the Enterprise server. The peer sever node on which a service has to be launched can be specified in the service's properties view. In the service's properties view, click on **Deployment** tab and provide the value of peer server node against the **Node** property.

When **CRC** is performed, the Enterprise server dynamically deploys the resources and dependencies of a service to the selected Fiorano Peer Server to launch and execute a service locally on the peer server

5.3 Run Configurations

1. After configuring the Event Process, right-click on the Event Process editor and select **Run As** → **Run Configurations...**

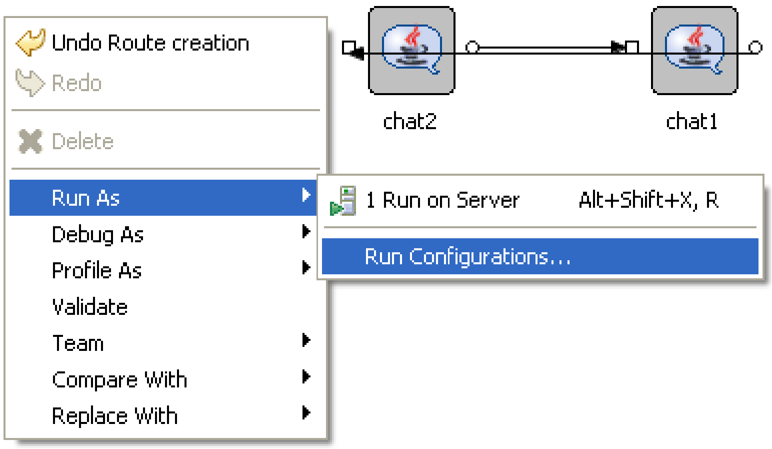
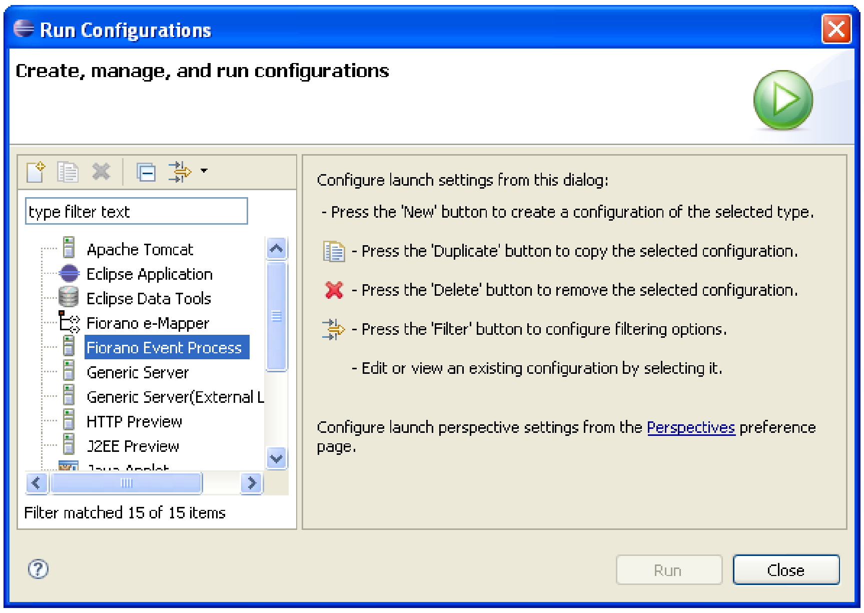


Figure 5.3.1: Run Configurations... option

2. The **Run Configurations** dialog box appears as shown in Figure 5.3.2.



Configure launch settings from this dialog:

- Press the 'New' button to create a configuration of the selected type.

- Press the 'Duplicate' button to copy the selected configuration.

- Press the 'Delete' button to remove the selected configuration.

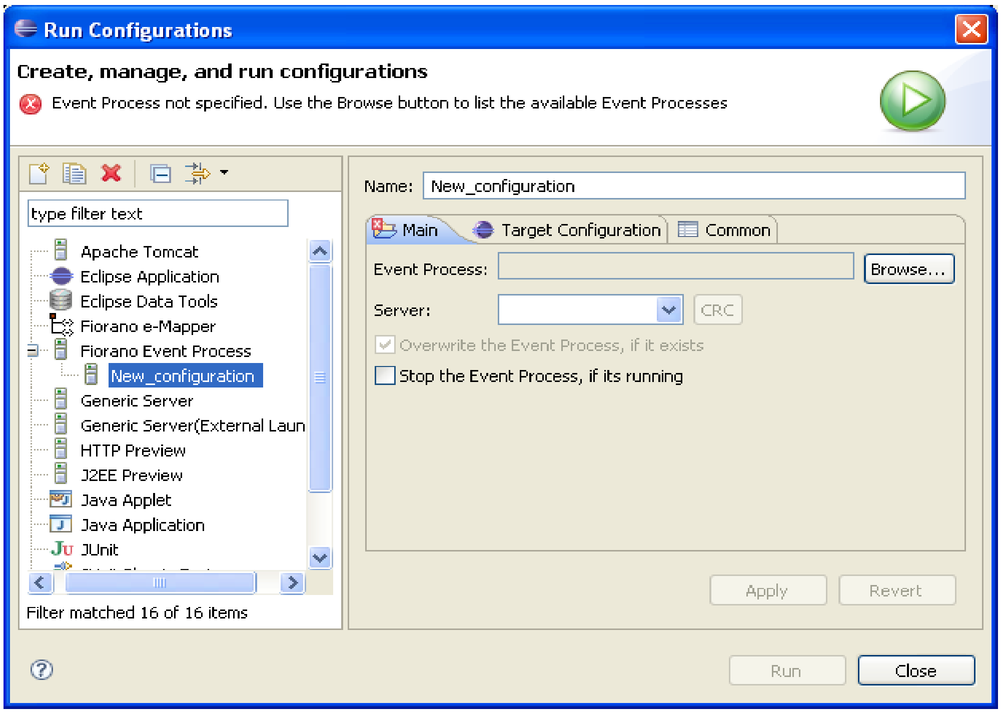
- Press the 'Filter' button to configure filtering options.

- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the Perspectives preference page.

Figure 5.3.2: Run Configurations dialog box

3. Double-click (or Right click and select New) on Fiorano Event Process icon to add a new configuration.



Event Process not specified. Use the Browse button to list the available Event Processes

Filter matched 16 of 16 items

Name: New_configuration

Event Process:

Server:

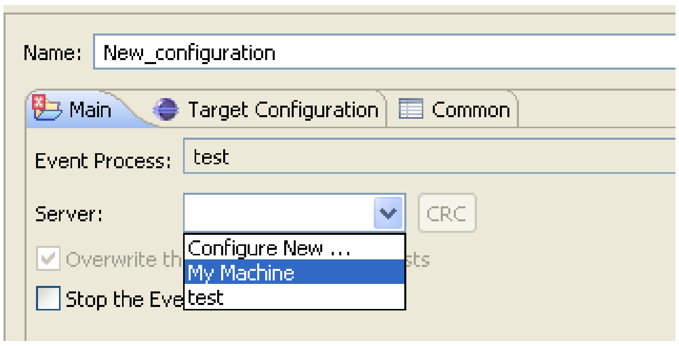
Overwrite the Event Process, if it exists

Stop the Event Process, if its running

Figure 5.3.3: Run Configuration dialog box

4. In **Run Configurations** window, click the **Browse** button to select the Event process to be launched.

5. Now, select the server on which the Event Process has to be launched.



Name: New_configuration

Event Process: test

Server:

Figure 5.3.4: Main tab

6. Select the appropriate server on which the Event Process has to be launched and click the **Apply** button in the dialog box to save the configuration.

5.4 Running Event Process

The target environment on which the Event process has to be launched can be selected.

Navigate to Run Configurations and click **Target Configuration** tab and select the appropriate environment as shown in Figure 5.4.1.

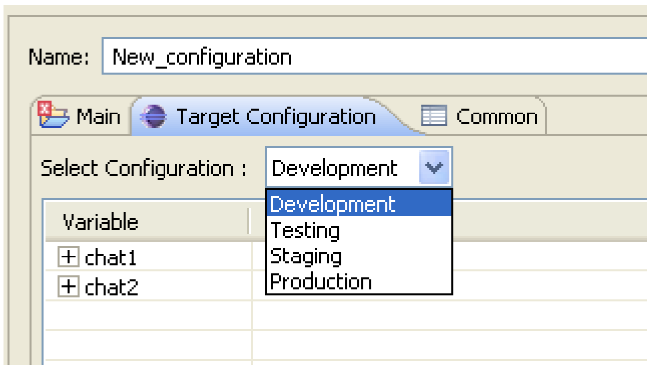


Figure 5.4.1: Target Configuration tab

5.5 CRC and Launch

After selecting the appropriate server in the Run Configurations window, CRC will be enabled. Click the **CRC** button to do Connectivity and Resource Check.

The Connectivity and Resource Check process involves the following:

- For each Component instance in the flow, checking if at least one of the peers in the deployment-node-list of the component instance is live and available within the Fiorano Network.
- Dynamically deploying the components and the external dependencies for the components used in the flow into the local repository of the peer on which the components runs.
- Checking to ensure that any two components with different schemas have an appropriate transformation defined if they are connected via a route.
- After the first three steps are successfully completed, Registration of the Event Process as **launchable** from the Fiorano Enterprise Server.

Thus at the click of a button and from a single point of control, services can be deployed on different peers across the enterprise network. After doing CRC, click the **Run** button to launch the Event Process.

5.6 Stopping Event Process

A running Event process can be stopped from the debug view. Select the configuration name in the Debug view and click the **Terminate** button to stop the Event Process.

Note: To stop a particular service, select the service and click the **Terminate** button.

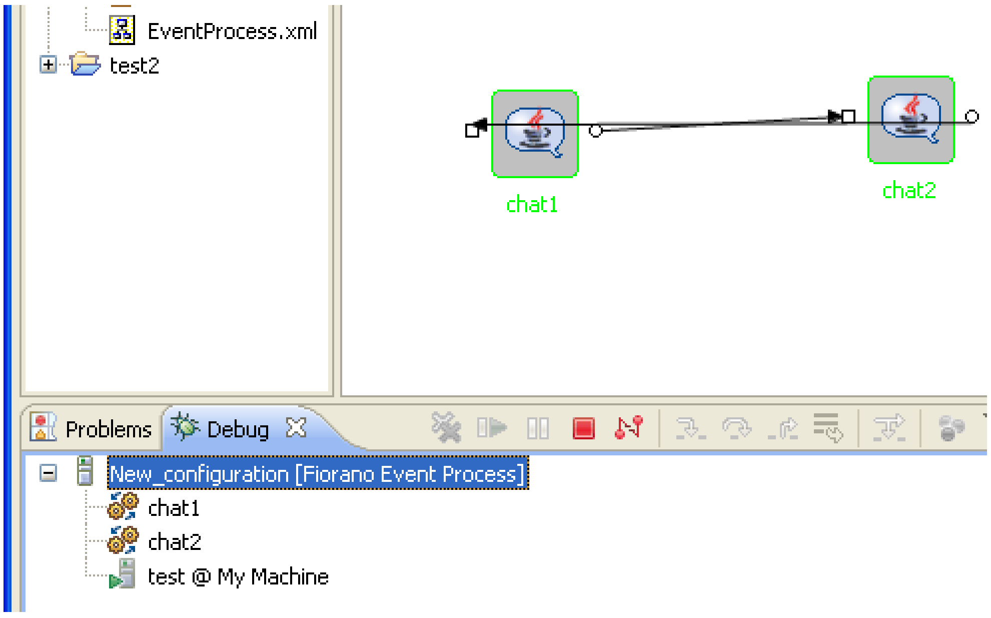


Figure 5.6.1: Stopping Event Process

Chapter 6: Debugging Event Process

To add a breakpoint to a route, open Fiorano Debugger View. Note that the option to add a breakpoint to a route is only available if the selected Event Process is running.

6.1 Adding Breakpoint

To add a breakpoint to a route, perform the following steps:

1. Click the **Add** button, the list of all the available routes for selected Event Process appears.
2. Click the route on which you want to add breakpoint and the click **OK**. The breakpoint will be added.

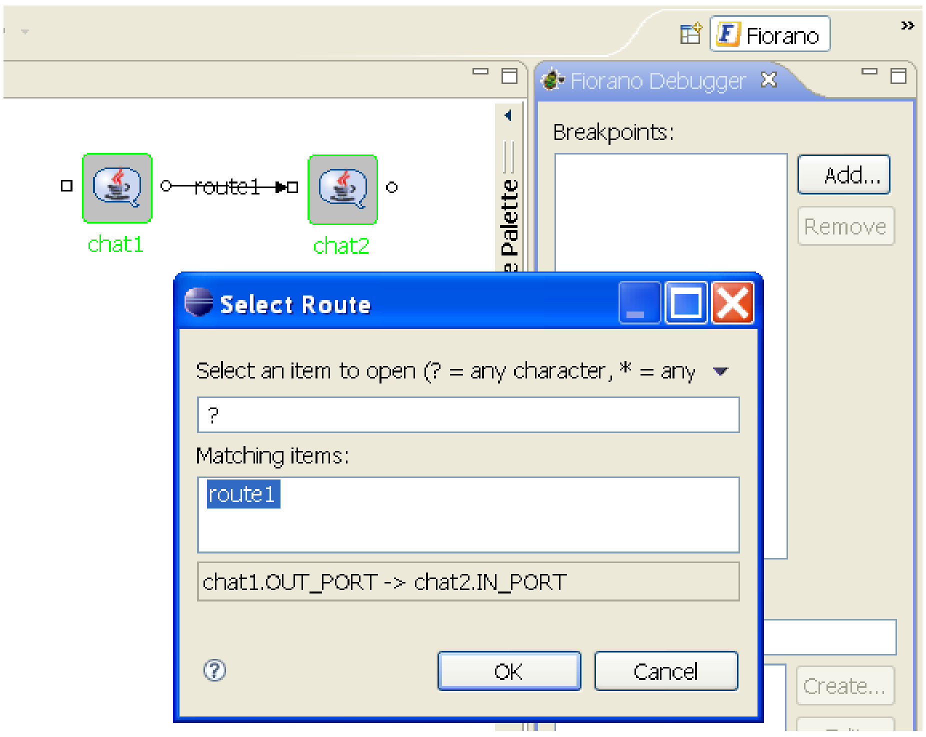
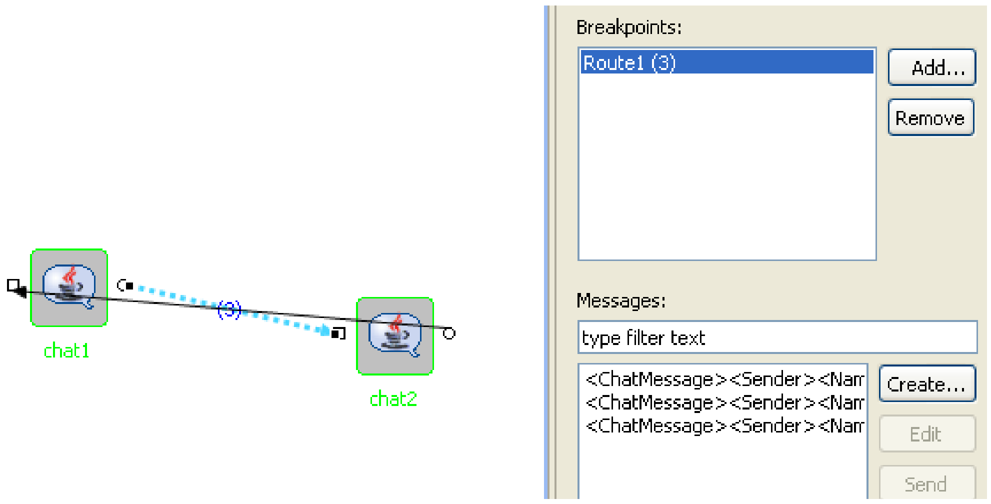


Figure 6.1.1: Select Route dialog box

6.2 Viewing Messages at Breakpoint

All the pending messages sent to a route which has breakpoint set on it are visible when clicked on that particular route in the breakpoint view.



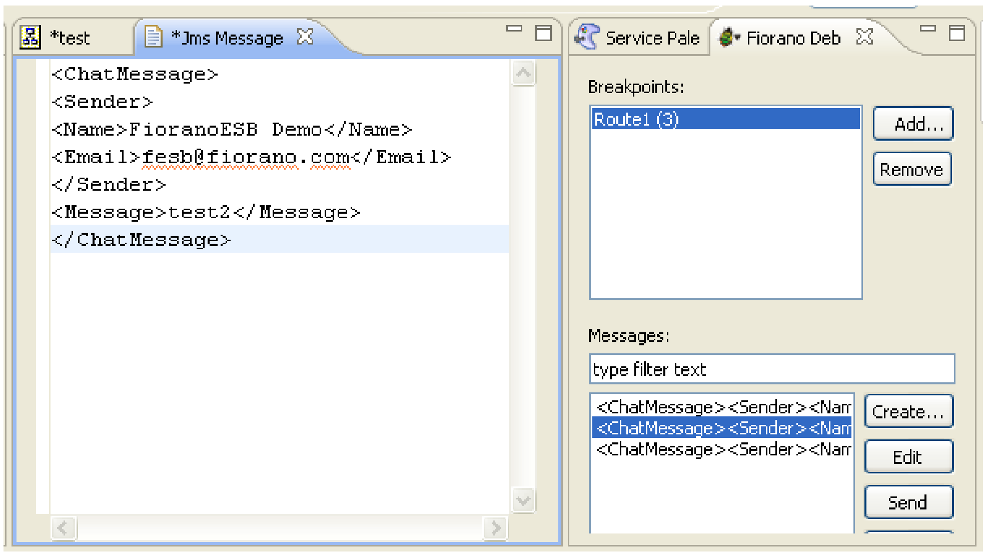
```
<ChatMessage><Sender><Nam
<ChatMessage><Sender><Nam
<ChatMessage><Sender><Nam
```

Figure 6.2.1: Message at breakpoint

6.3 Editing Messages at Breakpoint

To edit a message at debug time, perform the following steps:

1. Select the message which you want to edit and click the **Edit** button. The message will open up in an Editor where you can modify the message content.
2. Edit the message and click the **Save** button to save the message.



```
<ChatMessage>
<Sender>
<Name>FioranoESB Demo</Name>
<Email>fesb@fiorano.com</Email>
</Sender>
<Message>test2</Message>
</ChatMessage>
```

```
<ChatMessage><Sender><Nam
<ChatMessage><Sender><Nam
<ChatMessage><Sender><Nam
```

Figure 6.3.1: Save button

6.4 Inserting Messages into Breakpoint

You can choose to insert messages into breakpoint at debug time without the message being sent by the source component.

To insert messages into breakpoint, perform the following steps:

1. Click the **Create** button in the Messages view and choose the type of message, you want to create (either XML or Text message).

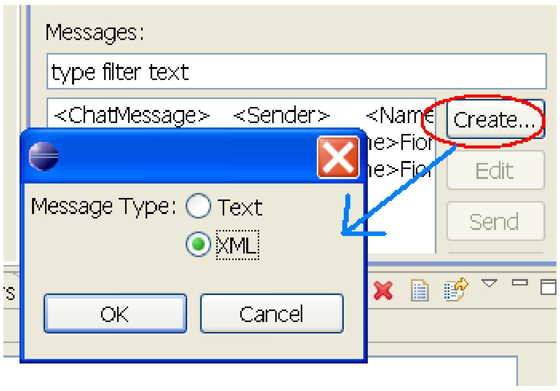


Figure 6.4.1: Selection of message type

2. Insert the content into the message content window that appears and save the content. The new message appears on the breakpoint in the Messages view.

6.5 Releasing Messages from Breakpoint

The messages present on a breakpoint can be released anytime so that they reach their destination.

To release the messages from the breakpoint, perform the following:

Select the message, you want to release and click the **Send** button. The sent message will now arrive at the destination port of the route selected.

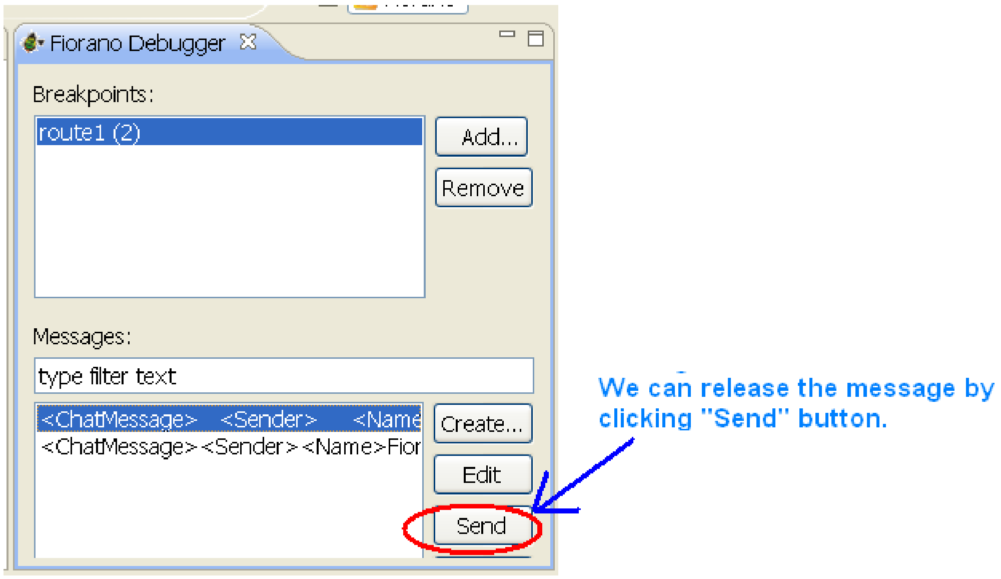

Fiorano Debugger
Breakpoints:
route1 (2)
Add...
Remove
Messages:
type filter text
<ChatMessage> <Sender> <Name
<ChatMessage><Sender><Name>Fior
Create...
Edit
Send
We can release the message by clicking "Send" button.

Figure 6.5.1: Fiorano Debugger

Chapter 7: Services

7.1 Service Descriptor Editor

1. A service can be customized using the Service Descriptor Editor. To customize a service, right-click the service in the Service Palette or in Service Repository view and select the **Edit…**. option.

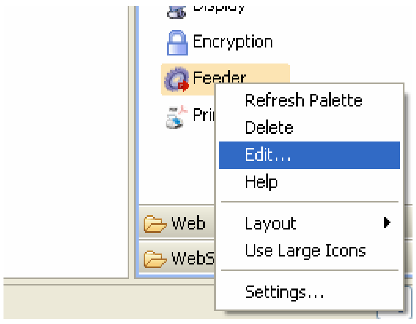
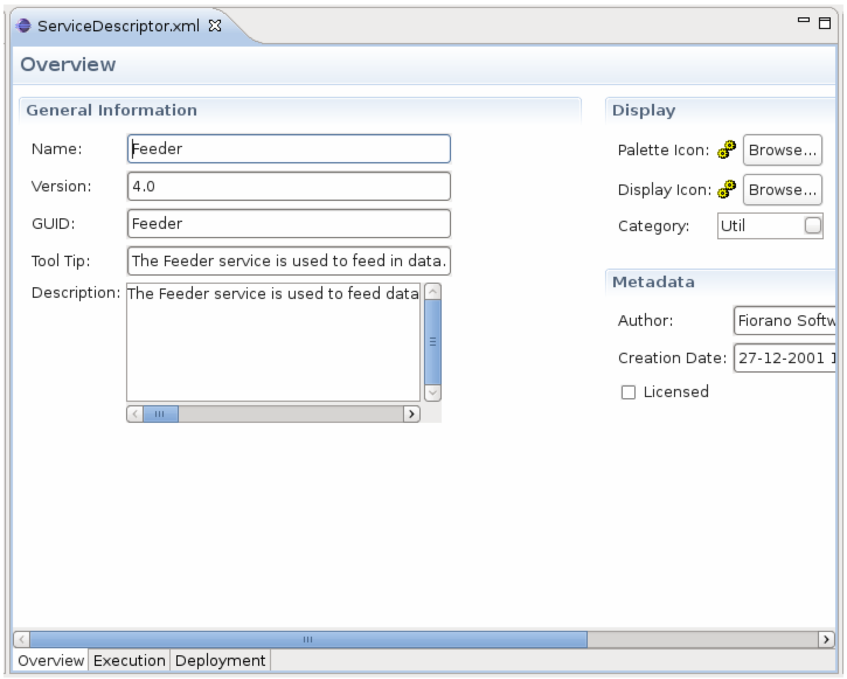


Figure 7.1.1: Edit option

2. The **ServiceDescriptor.xml** of the selected service is opened in the Service Descriptor Editor as shown in Figure 7.1.2.


ServiceDescriptor.xml

Overview

General Information

Name: Feeder

Version: 4.0

GUID: Feeder

Tool Tip: The Feeder service is used to feed in data.

Description: The Feeder service is used to feed data

Display

Palette Icon: Browse...

Display Icon: Browse...

Category: Util

Metadata

Author: Fiorano Softw

Creation Date: 27-12-2001 1

☐ Licensed

Overview | Execution | Deployment

Figure 7.1.2: ServiceDescriptor.xml

Service Descriptor Editor has three pages, they are:

- Overview
- Execution
- Deployment

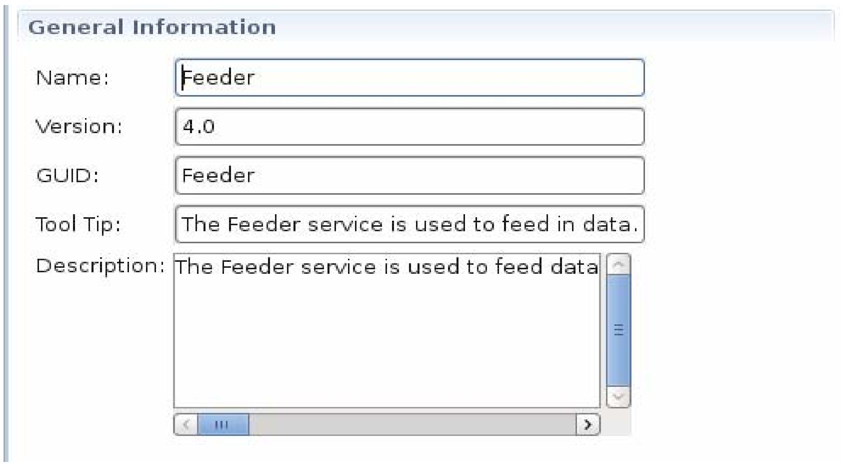
These pages are further divided into different sections. A brief explanation of these pages and subsections is provided below.

The pages can be accessed using the tabs provided at the bottom left corner of the editor.

7.1.1 Overview Page

The Overview page has three subsections – General Information, Display, and Metadata.

The information used to identify the service is shown under **General Information** section. The user can change the Name, Version, GUID, Tool Tip, and Description of the component.

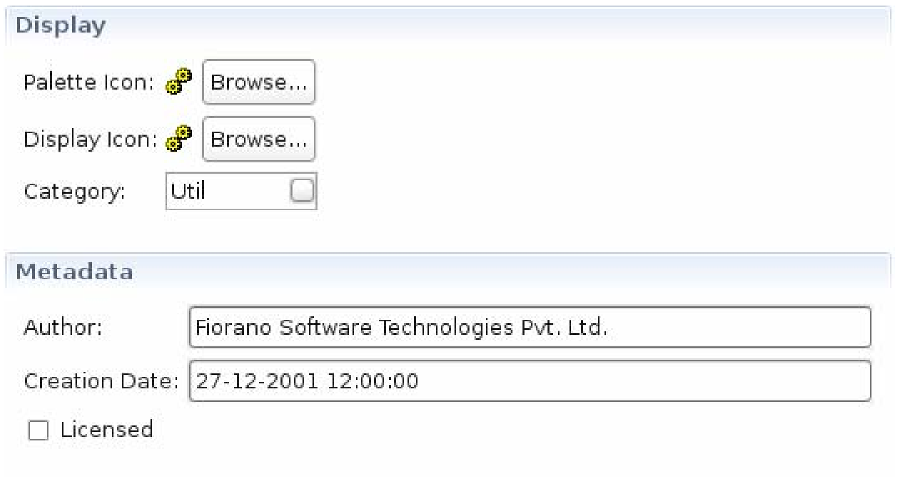

General Information

Name:	Feeder
Version:	4.0
GUID:	Feeder
Tool Tip:	The Feeder service is used to feed in data.
Description:	The Feeder service is used to feed data

Figure 7.1.3: General Information

In the **Display section**, the icons used to represent the service, and the categories under which the service is placed are provided. Categories can be selected using the Category Selection dialog box, which is similar to the one used during Service Creation.

In the **Metadata section**, the information about authors of the service, creation date and time of the service and licensing mode are provided.


Display

Palette Icon:	Browse...
Display Icon:	Browse...
Category:	Util

Metadata

Author:	Fiorano Software Technologies Pvt. Ltd.
Creation Date:	27-12-2001 12:00:00

☐ Licensed

Figure 7.1.4: Metadata section

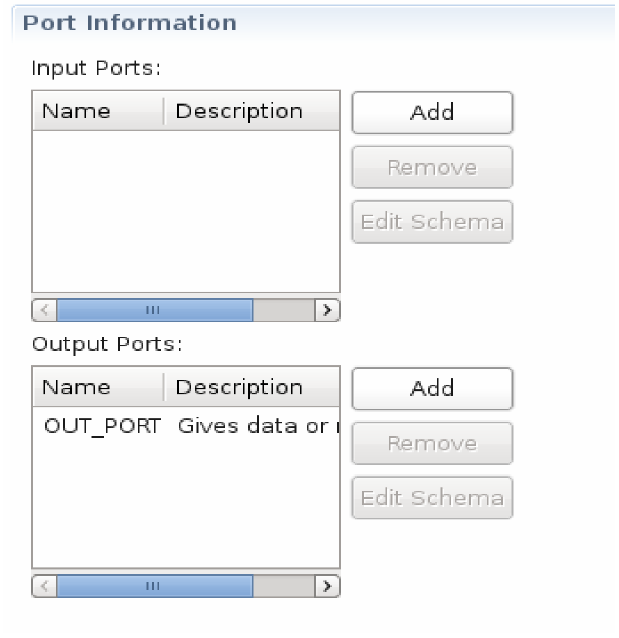
7.1.2 Execution Page

Execution Page has the following subsections – Port Information, Support, Launch Configuration, Log Modules, and Runtime. A brief explanation of these subsections is provided below.

Port Information

Each Asynchronous Service Component (also referred as **Event Driven Business Component**) can have any number of inputs and outputs as determined by the developer of the component. Input and output ports can be added or removed in the Service Descriptor Editor as applicable to the component in **Port Information** section.

The Add, Remove and Edit Schema buttons can be used to add, remove and to edit on ports of service. Name, Description of any port can be modified from the respective columns in each table.

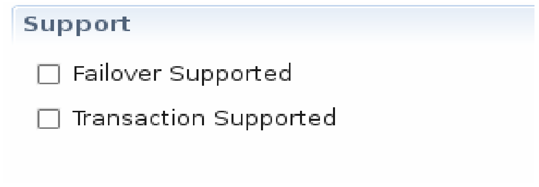


Name	Description
OUT_PORT	Gives data or

Figure 7.1.5: Port Information section

Support

In the **Support** section, **Failover Supported** and **Transaction Supported** options are present.



☐ Failover Supported
☐ Transaction Supported

Figure 7.1.6: Support section

- Failover Supported

  If the Failover Supported option is selected, during component's runtime if the Peer Server on which component is running goes down, the component keeps running on the next available PeerServer.

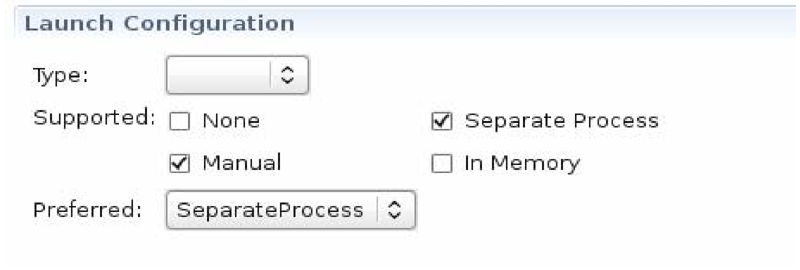
  If this option is not selected, at component's runtime, if the Peer Server on which component is running goes down, the component stops.

- Transaction Supported

  Transaction Supported is used to specify whether the service allows transacted session or not.

Launch Configuration

In the **launch configuration** section, information about type of the component and the different launch types supported (Separate Process, InMemory, Manual) are provided.



Launch Configuration

Type:

Supported: [ ] None [✓] Separate Process [✓] Manual [ ] In Memory

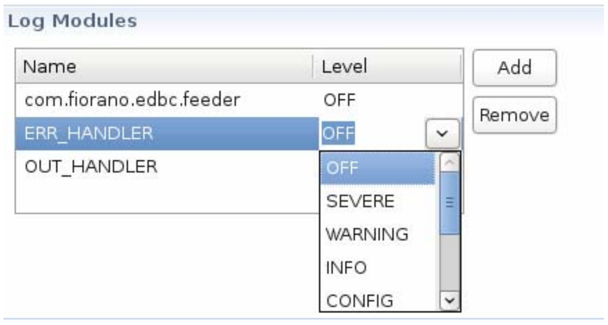
Preferred: SeparateProcess

Figure 7.1.7: Launch configuration section

Log Modules

In the **Log Modules** section logging options of the service are provided. Loggers used to log messages during service runtime can be added or removed.

To add a new logger click **Add** and specify log module name and the log level at which logging has to be performed. Messages logged at level lower than the selected log level will not be written to log files.



Log Modules

Name	Level
com.fiorano.edbc.feeder	OFF
ERR_HANDLER	OFF
OUT_HANDLER	

Figure 7.1.8: Log Modules section

Runtime

In the **Runtime** section configurations required to launch services are provided.

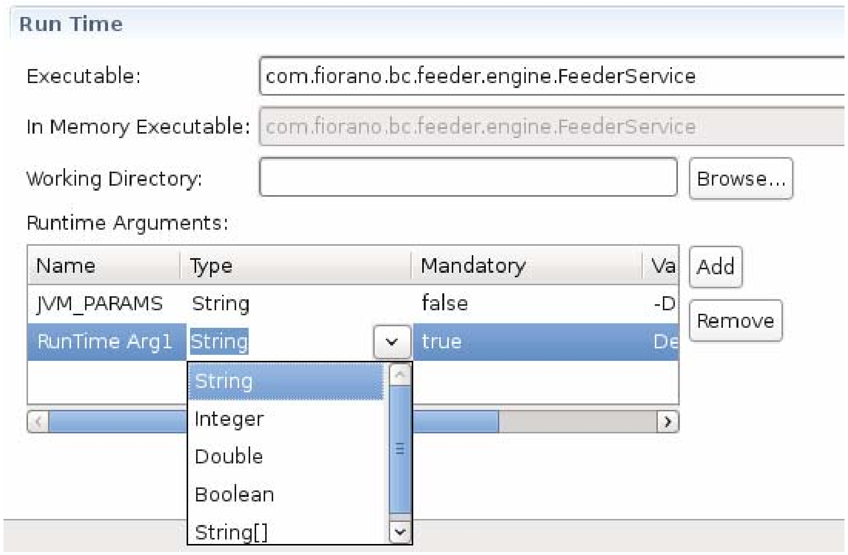
Executable specifies the Java class to be used to launch the service when it is launched in a Seperate Process

In Memory Executable specifies the Java class to be used when the service is launched in in-memory mode.

Working directory specifies the directory which will be service's runtime directory when launched in separate process.

A component while executing, might require parameters to execute different requests or details of handling different request. There are two ways of passing this information to the component. By configuring the details in the Configuration Property Sheet of the panel.

By defining the command line arguments that can be passed to the component during the launch of the component. These command line arguments are captured as runtime arguments in this panel.



Run Time

Executable: com.fiorano.bc.feeder.engine.FeederService

In Memory Executable: com.fiorano.bc.feeder.engine.FeederService

Working Directory:

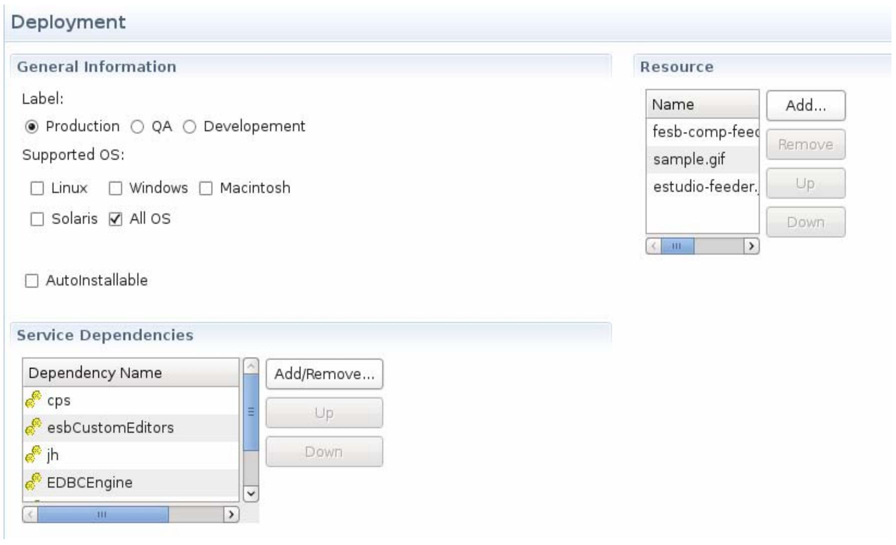
Runtime Arguments:

Name	Type	Mandatory	Va
JVM_PARAMS	String	false	-D
RunTime Arg1	String	true	De

Figure 7.1.9: Runtime section

7.1.3 Deployment Page

This page contains subsections related to deployment information of the component. Resources/Service Dependencies required by the component can be configured in this page.



Deployment

General Information

Label: [✓ Production] QA Developement

Supported OS: Linux Windows Macintosh Solaris [✓ All OS]

AutoInstallable

Resource

Name
fesb-comp-feec
sample.gif
estudio-feeder.

Service Dependencies

Dependency Name
cps
esbCustomEditors
jh
EDBCEngine

Figure 7.1.10: Deployment page

Resource

The resources required by the service (either during configuration time or runtime) can be added in this wizard. Resources can be any files which are used by the component. Typically resource files are of types – dll, zip, jar, so, exe.

- To add a resource click on **Add...** and select required resource for the service.
- To remove a resource, select the resource and click **Remove.**
- To change the order of resources, select the resource and click **Up** or **Down**. The order is used to determine the classpath of the service

Service Dependencies

Dependencies are predefined. Every component or system library registered can be added as a dependency.

Click the **Add/Remove** button to open **Add Dependencies** dialog box. It has a list of all available dependencies.

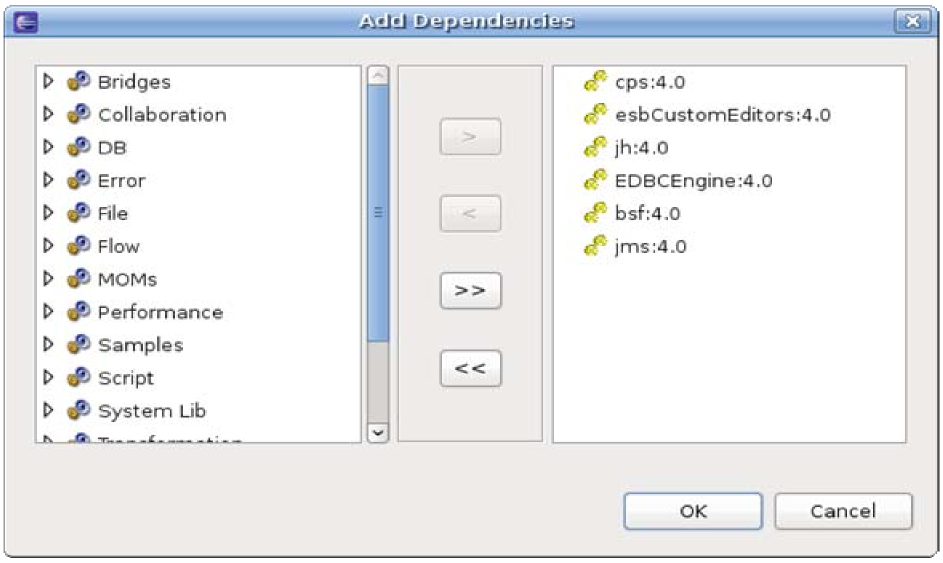


Figure 7.1.11: Service Dependencies section

To Add: Select the dependency on left side table of the page and move to the right side table.

To Remove: Select the dependency on right side table of the page and move to the left side table.

7.2 Service Repository

Fiorano eStudio has an independent service repository which enables services to be configured offline (without connection to Enterprise Server).

The service repository can be viewed by opening the Service Repository view which displays categorized services as shown in Figure 7.2.1.

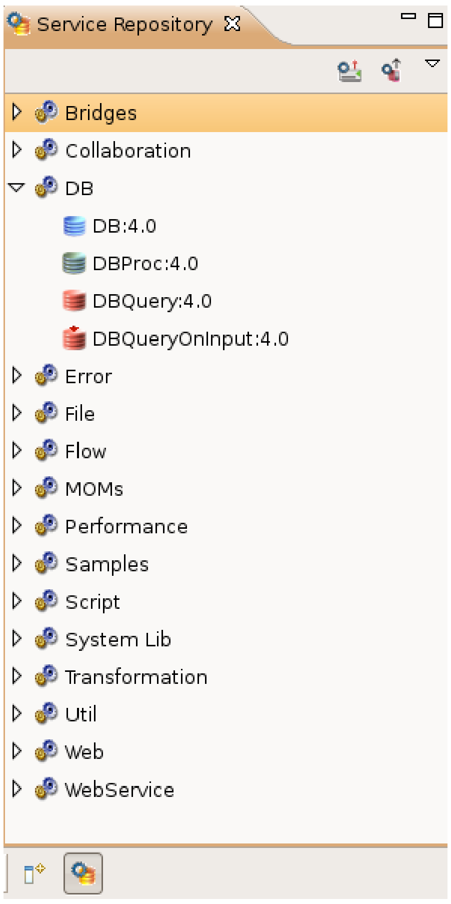
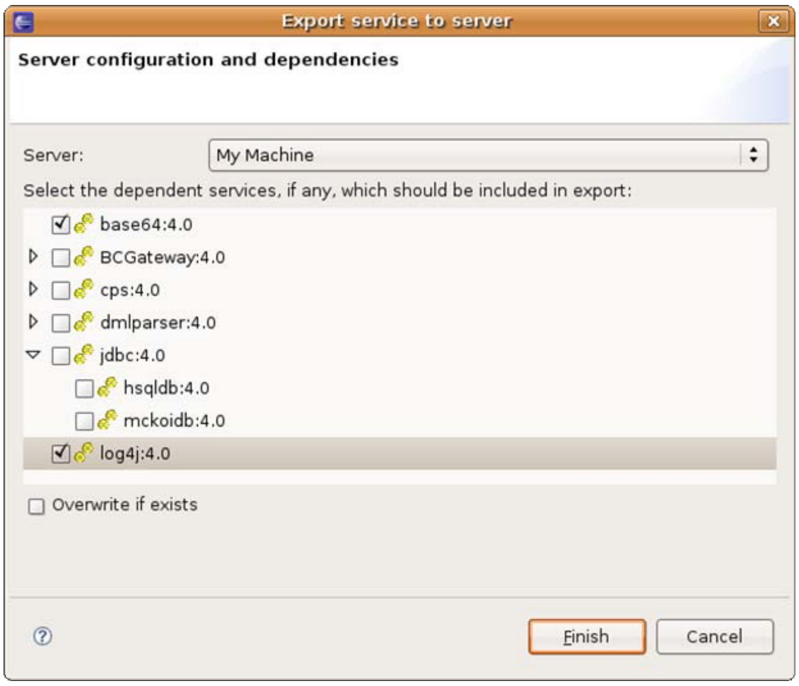


Figure 7.2.1: Service repository

7.2.1 Deploying Services to Server

A service can be deployed to an Enterprise server by right-clicking the component in service repository view and selecting Export to Server from the context menu.

This opens up a dialog as shown in Figure 7.2.2.



Export service to server

Server configuration and dependencies

Server: My Machine

Select the dependent services, if any, which should be included in export:

- [✓] base64:4.0
- [ ] BCGateway:4.0
- [ ] cps:4.0
- [ ] dmlparser:4.0
- [ ] jdbc:4.0
  - [ ] hsqldb:4.0
  - [ ] mckoidb:4.0
- [✓] log4j:4.0

[ ] Overwrite if exists

Figure 7.2.2: Export service to server

The dialog shows a dependency tree excluding the actual service (which gets exported by default). To export any dependencies of this service, they too have to be selected from the tree.

If **Overwrite if exists** checkbox is selected, service in the server will be over written by the one in service repository, otherwise conflicting services are not exported to server.

7.2.2 Fetching Services from Server

The services present on server can be imported into the service repository by selecting the option **Import from Server** as shown in Figure 7.2.3.

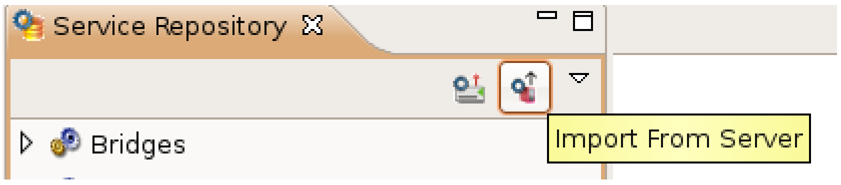
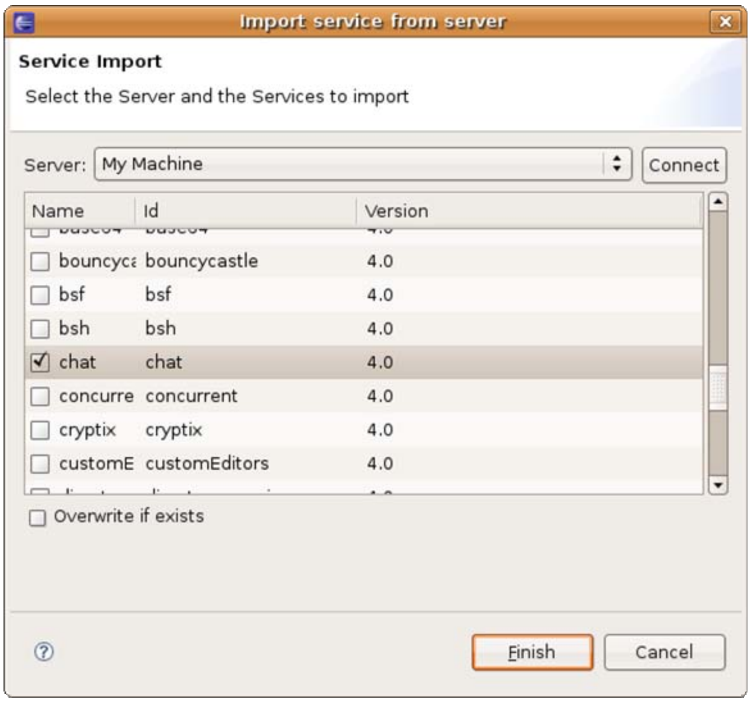


Figure 7.2.3: Import from Server option

This opens **Import service from server** dialog box as shown in Figure 7.2.4.


Import service from server

Service Import

Select the Server and the Services to import

Server: My Machine

Name	Id	Version
☐ bouncyca	bouncycastle	4.0
☐ bsf	bsf	4.0
☐ bsh	bsh	4.0
✓ chat	chat	4.0
☐ concurre	concurrent	4.0
☐ cryptix	cryptix	4.0
☐ customE	customEditors	4.0

☐ Overwrite if exists

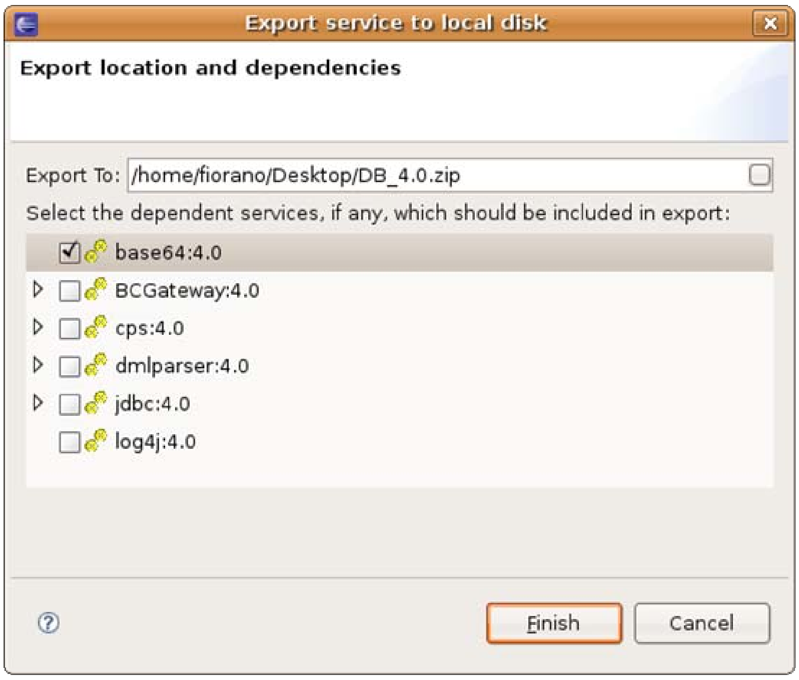
Figure 7.2.4: Import service from server

The selected services are imported from server to the eStudio service repository.

If **Overwrite if exists** checkbox is selected, service in the service repository will be over written by the one in server, otherwise conflicting services are not imported from server.

7.2.3 Exporting Services to Local Disk

Services in Service Repository can be exported to local disk by right-clicking the service and selecting **Export service to local disk** option from context menu. This opens a dialog box as shown in Figure 7.2.5.


Export service to local disk
Export location and dependencies

Export To: /home/fiorano/Desktop/DB_4.0.zip

Select the dependent services, if any, which should be included in export:

- [✓] base64:4.0
- [ ] BCGateway:4.0
- [ ] cps:4.0
- [ ] dmlparser:4.0
- [ ] jdbc:4.0
- [ ] log4j:4.0

Figure 7.2.5: Export service to Local Disk

The export location can be chosen by the user. By default the selected service gets included in the export. To export dependent services, they have to be selected from the tree as shown in Figure 7.2.5.

7.2.4 Importing Services from Local disk

The components can be imported from the file system. This can be done by selecting **Import from Local Disk** button as shown in Figure 7.2.6.

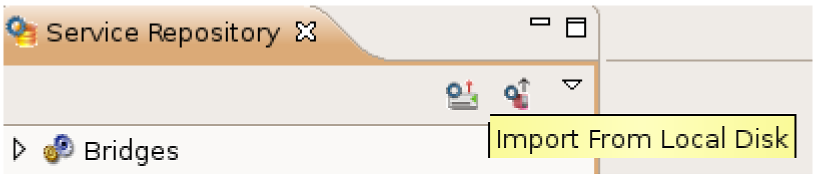
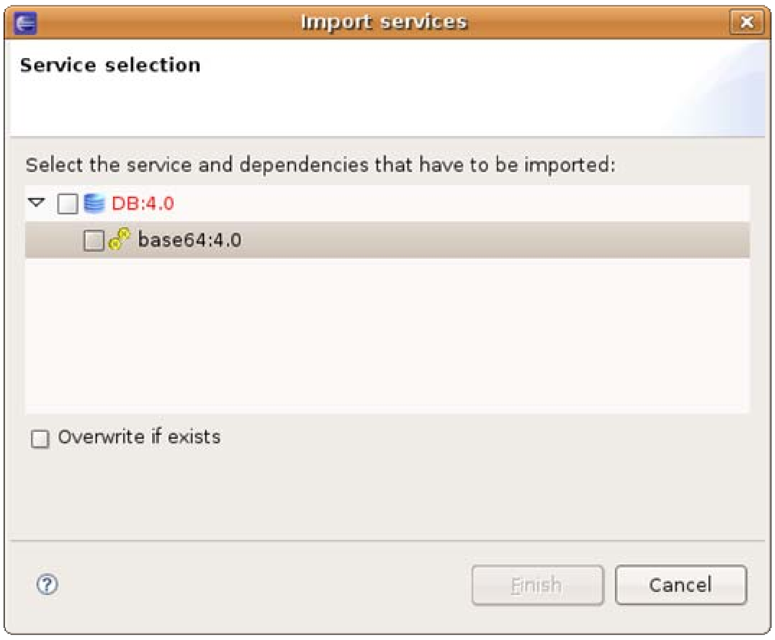


Figure 7.2.6: Import from Local Disk button

This opens a file selection dialog box with which the zip file containing services on the disk is selected. Upon selection a dialog box is shown in which the services in the zip file are shown in a dependency tree as shown in Figure 7.2.7.



Select the service and dependencies that have to be imported:

- ☐ DB:4.0
  - ☐ base64:4.0

☐ Overwrite if exists

Figure 7.2.7: Import services dialog box

Components which are already present in the server are labeled red, and those not present in repository are labeled black.

If **Overwrite if exists** checkbox is selected, service in the service repository will be over written by the one in the zip file, otherwise conflicting services are not imported from local disk.

Chapter 8: Service Creation

Apart from the exhaustive list of pre-built services, custom services can be written, built, and deployed into Fiorano SOA Platform by developers. To aid developers in service creation, the platform provides a template engine to generate the skeleton code for custom services in Java, C, C++, C# (.Net)

8.1 Service Generation

To create a new service - open Fiorano Perspective and select **Tools** -> **Create Service Component** to open the Service Creation Wizard. All the details related to the creation of a new service must be specified in this wizard. Various steps in service creation are illustrated below.

8.1.1 Service Location

The destination folder in which the component has to be created has to be specified.

Note: A new folder name has to be specified here. If the folder name provided already exists, then the wizard does not allow proceeding to the next page.

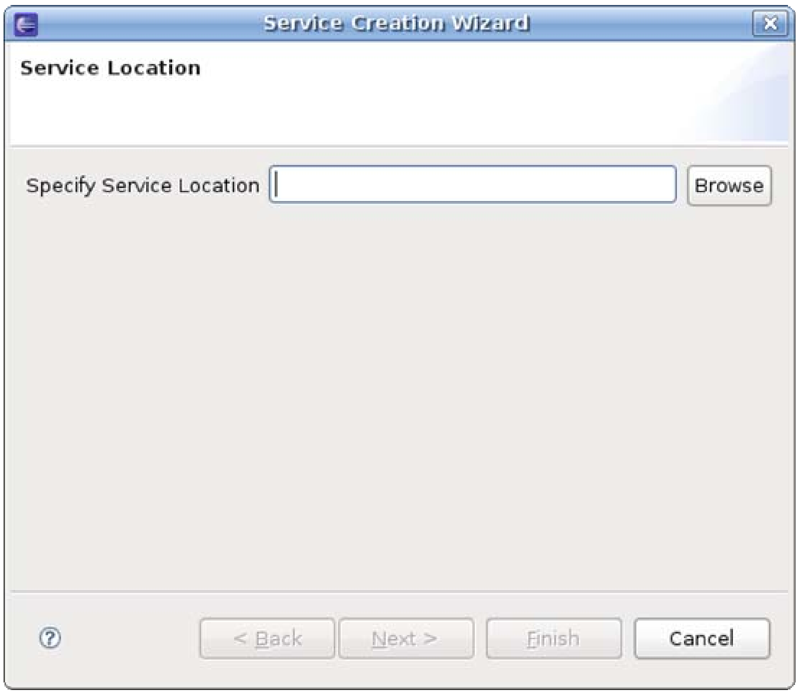
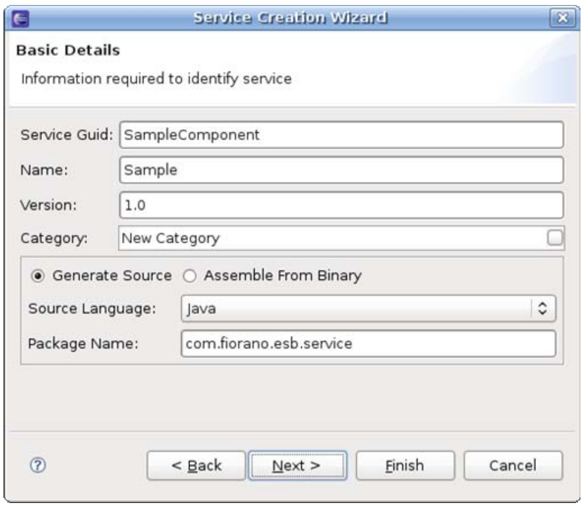


Figure 8.1.1: Specific Service Location

8.1.2 Basic Details

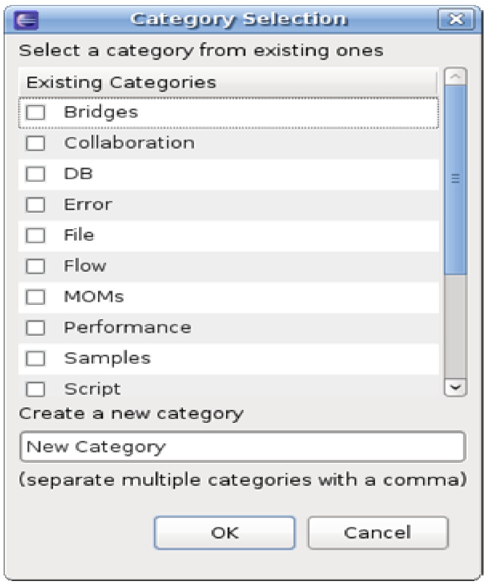
The Basic Details of the service like Service Guid, Name, Version and so on have to be provided here.



Service Guid: SampleComponent
Name: Sample
Version: 1.0
Category: New Category
[✓ Generate Source] Assemble From Binary
Source Language: Java
Package Name: com.fiorano.esb.service

Figure 8.1.2: Service creation wizard

In the Category field, a new Category name can be provided for the component or an existing Category can be selected from the available categories. Existing Categories can be viewed by clicking the ellipsis button against the Category field. On clicking ellipsis the Category Selection dialog box appears as shown in Figure 8.1.3. Multiple Categories can also be selected in the Category Selection dialog box.



Select a category from existing ones

Existing Categories
- ☐ Bridges
- ☐ Collaboration
- ☐ DB
- ☐ Error
- ☐ File
- ☐ Flow
- ☐ MOMs
- ☐ Performance
- ☐ Samples
- ☐ Script

Create a new category

New Category

(separate multiple categories with a comma)

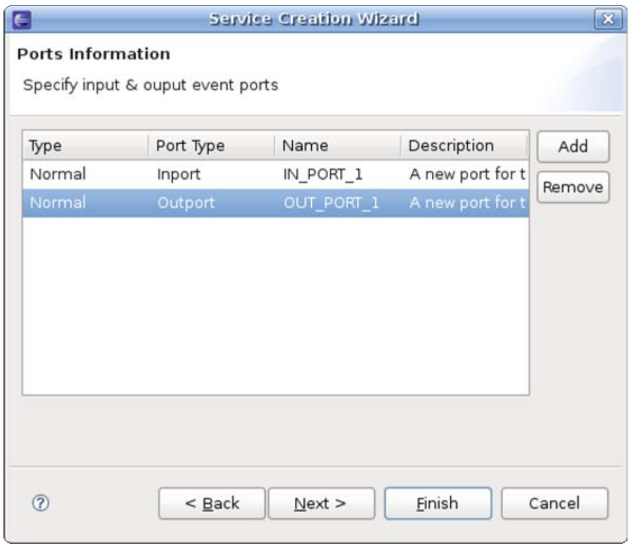
Figure 8.1.3: Category Selection dialog box

The option **Generate Source** is used to generate sources for various languages and the option **Assemble From Binary** is used to create System Libraries.

8.1.3 Ports Information

The input and output ports of the service can be configured here.

A new port can be added by clicking the **Add** button. By default Port Type is Input Port. Port Type and other port properties can be changed in the wizard as required.



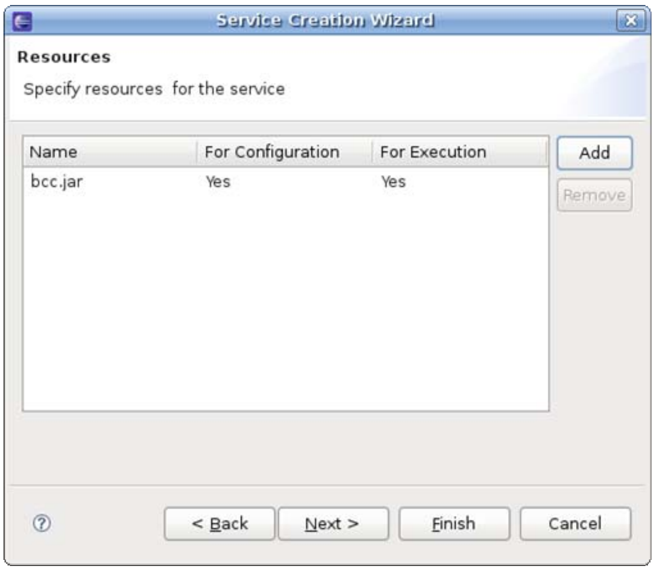
Type	Port Type	Name	Description
Normal	Inport	IN_PORT_1	A new port for t
Normal	Outport	OUT_PORT_1	A new port for t

Figure 8.1.4: Ports Information

8.1.4 Resources

The resources required by the service (either during configuration time or runtime) can be added in this wizard. Resources can be any files which are used by the component. Typically resource files are of types – dll, zip, jar, so, exe. However, there is no strict restriction on this, a file of any type can be added as a resource.

The server makes a local copy of these files in the component's folder. Resources can be added or removed using **Add** and **Remove** buttons.



Name	For Configuration	For Execution
bcc.jar	Yes	Yes

Figure 8.1.5: Resources section

8.1.5 Dependencies

Dependencies are predefined. Every component or system library registered can be added as a dependency. The dependencies are referenced from the existing location and are not copied locally into the component's folder.

Note: Dependencies are loaded only once when the components are launched in-memory of same peer server, there by reducing the memory footprint.

To Add: Select the dependency on left-hand side of the page and move to the right-hand side.

To Remove: Select the dependency on right-hand side of the page and move to the left-hand side.

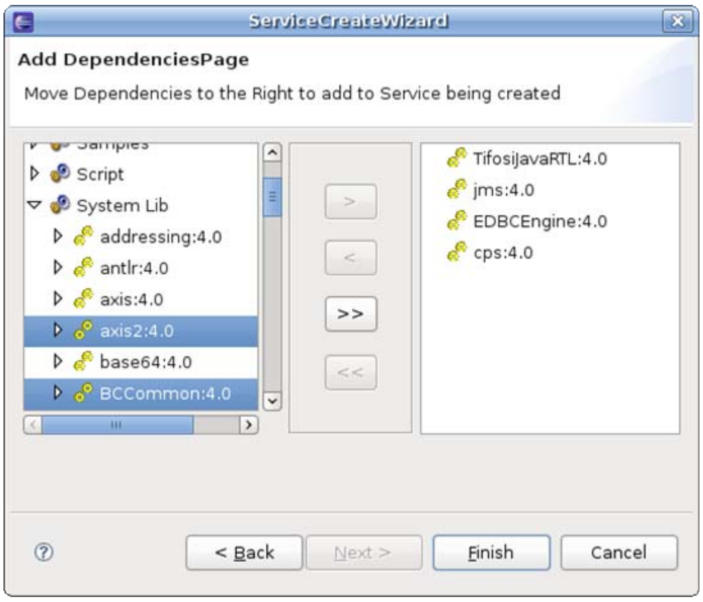


Figure 8.1.6: Dependencies

Click the **Finish** button after adding the dependencies.

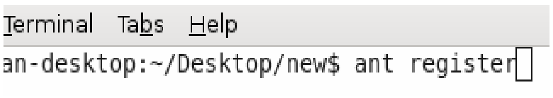
When the wizard is finished, sources are generated under **src** directory in the directory specified in the Service Location Page. It also creates necessary files to build and deploy the components.

8.2 Building and Deploying Services

By default **build.properties** file contains the URL of the Enterprise Server running on the machine on which the sources are generated. If the service has to be deployed to an Enterprise Server running on a different machine then the property server has to be changed in the **build.properties** file.

To register the service, perform the following steps:

1. Open the command prompt at the location where the sources are generated and execute the command **ant register**.



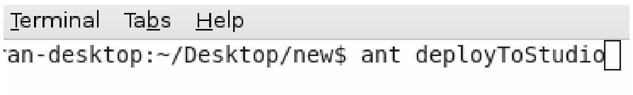
```
an-desktop:~/Desktop/new$ ant register
```

Figure 8.2.1: ant register

2. This builds the service's sources and registers the service with the Enterprise Server.

To view the service in eStudio service palette, perform the following steps:

1. Execute the command ant **deployToStudio**.



```
an-desktop:~/Desktop/new$ ant deployToStudio
```

Figure 8.2.2: deployToStudio

2. The service is now available in eStudio Service Palette and can be used in composing EventProcesses.

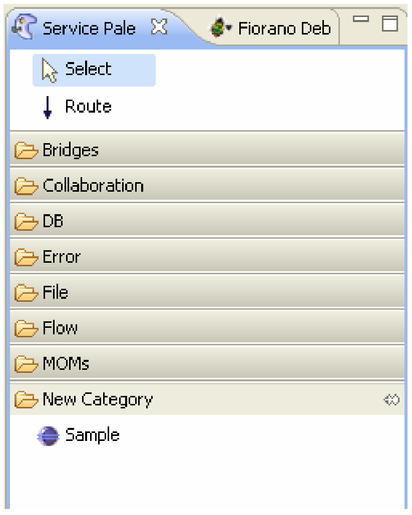


Figure 8.2.3: Service Palette

Chapter 9: Mapper

9.1 Creating Mapper Project

1. From the File menu, select **New** -> **Fiorano Mapper Project**.

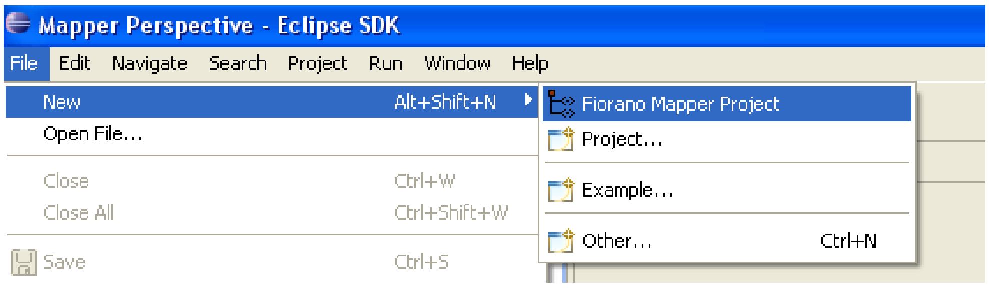
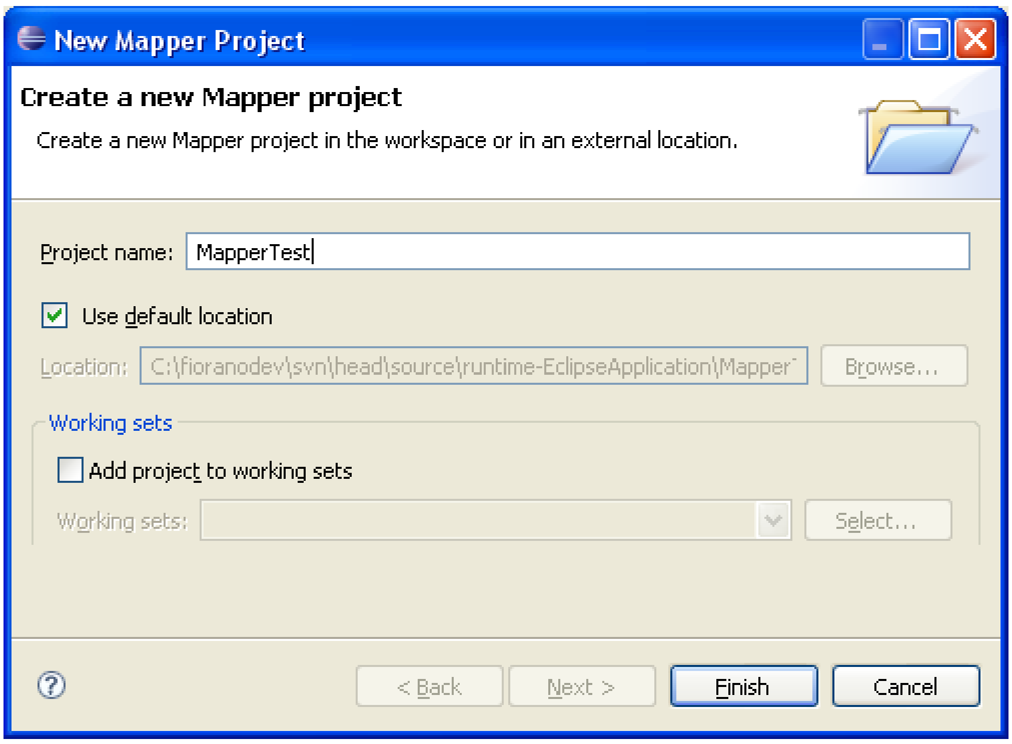


Figure 9.1.1: Fiorano Mapper Project option

2. Specify the name of the Fiorano Mapper project in the **Project Name** field and click the **Finish** button.


New Mapper Project

Create a new Mapper project

Create a new Mapper project in the workspace or in an external location.

Project name: MapperTest

[✓] Use default location

Location: C:\fioranodev\svn\head\source\runtime-EclipseApplication\Mapper

Working sets

[ ] Add project to working sets

Working sets:

Figure 9.1.2: Finish button

3. A new project with specified name appears in Project Explorer and a new editor `MapperTest.fmap` is shown. If the editor is not shown, expand the project created in project explorer and double-click the `MapperTest.fmap`

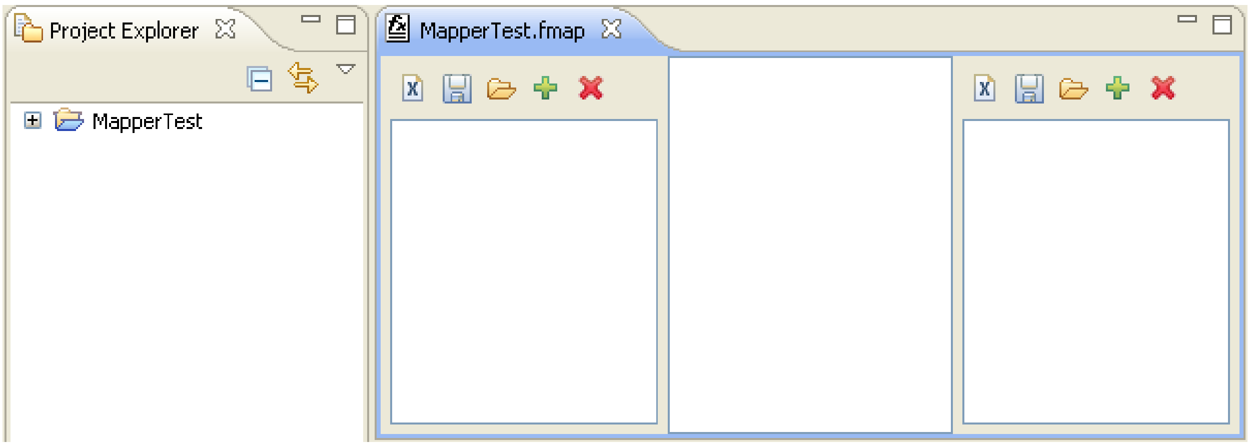
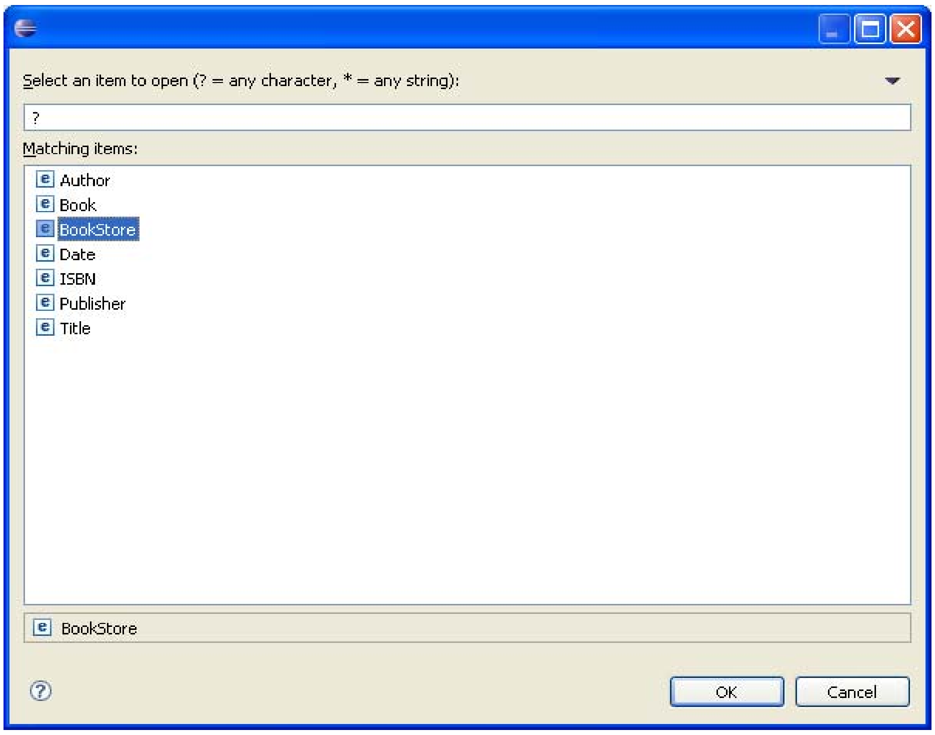


Figure 9.1.3: MapperTest.fmap

1. To add a structure (input/output), click the (Add Structure) button.
2. Select a file to open. The file types xsd, dtd and xml can be selected.
3. When the selected file contains multiple possible root elements a list of all root elements is shown. Select required root element and click the **OK** button.


Select an item to open (? = any character, * = any string):

?

Matching items:

Author
Book
BookStore
Date
ISBN
Publisher
Title

BookStore

Figure 9.1.4: Ok button

4. Load input and output structures by repeating steps 4 to 6.
5. Define mappings by selecting an element or attribute of input structure, dragging and dropping on an element or attribute of output structure.

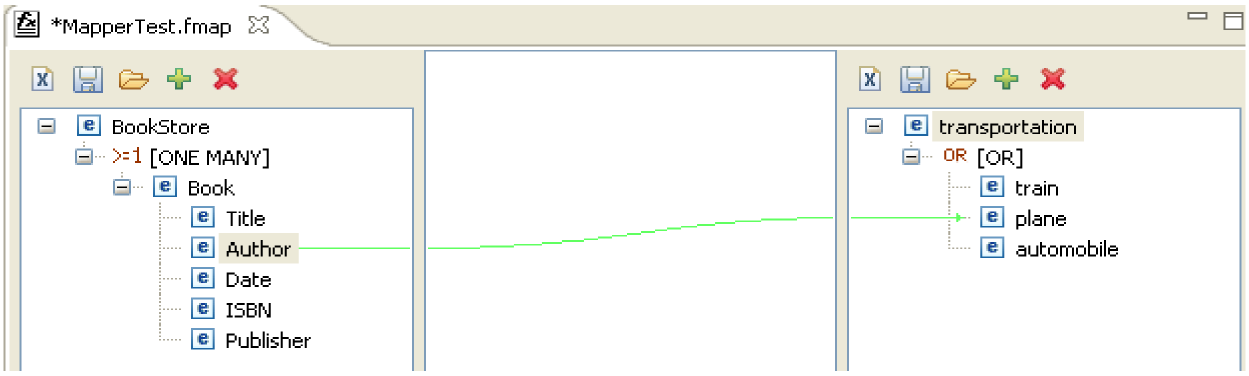


Figure 9.1.5: Attribute of output structure

6. To enrich the mapping with functions, select the mapping. Selected mapping will be shown in **Funclet View**.

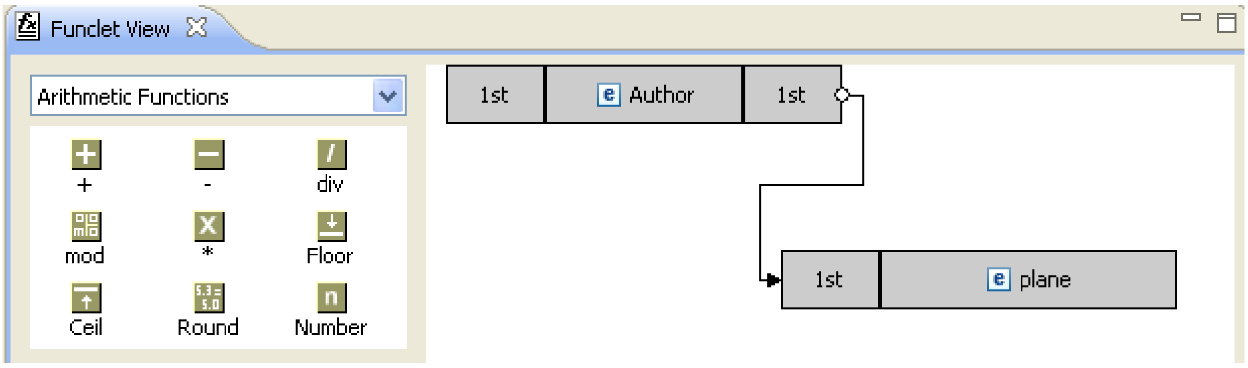


Figure 9.1.6: Funclet View

7. Select required functions category from the drop-down list.

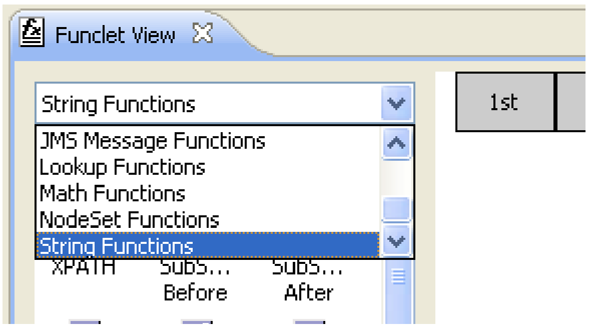


Figure 9.1.7: Function category

8. Select function of choice. Drag and drop the function into mapping area and connect input and output nodes with the function selected. Multiple input nodes may be selected (drag-drop).

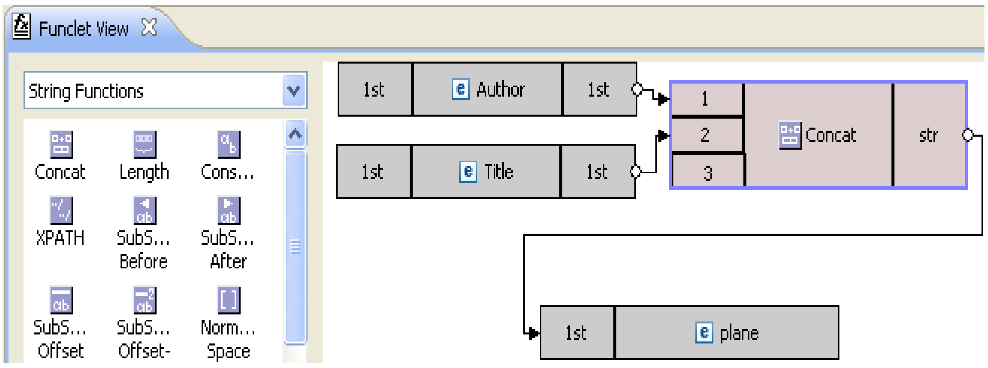


Figure 9.1.8: Input nodes

9. Save the project. XSL is created in the project at the following location **resources/xsl/<project_name>.xsl**.

9.2 Testing Transformation

To test the transformation defined, perform the following steps:

1. From the **Run** menu, choose **Run Configurations** option.

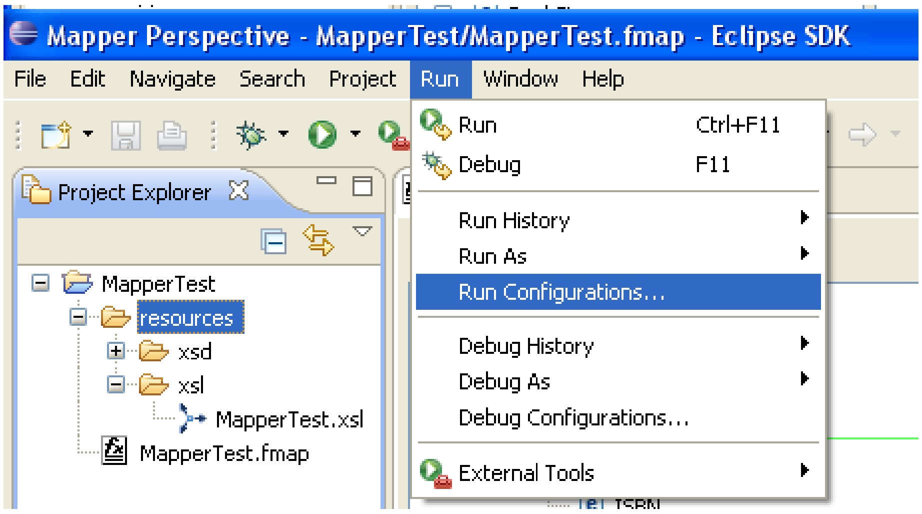


Figure 9.2.1: Run Configurations... option

2. In the **Run Configurations** right-click on **Fiorano e-Mapper** and click **New** in the pop-up menu.

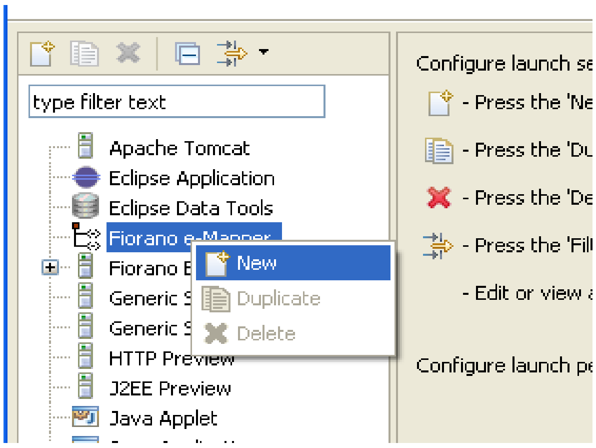


Figure 9.2.2: New option

3. Click **Browse...** for **Fmap File** and select the required .**fmap** from the project.
4. Provide an input xml file which has to be transformed at **Input xml File**.
5. Alternatively, check **Generate From XSD** and click **Generate XML** to generate a sample input file.
6. Select the location of sample XML file and click the **Next** button.

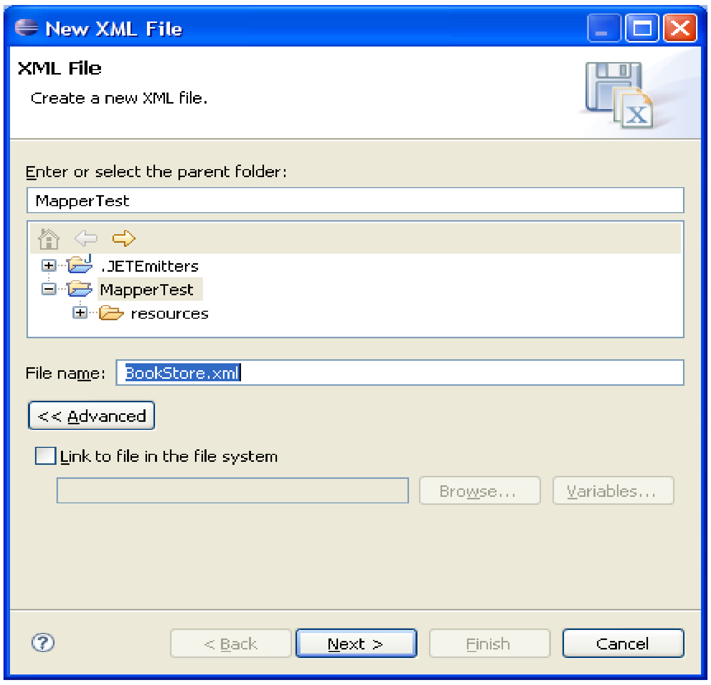
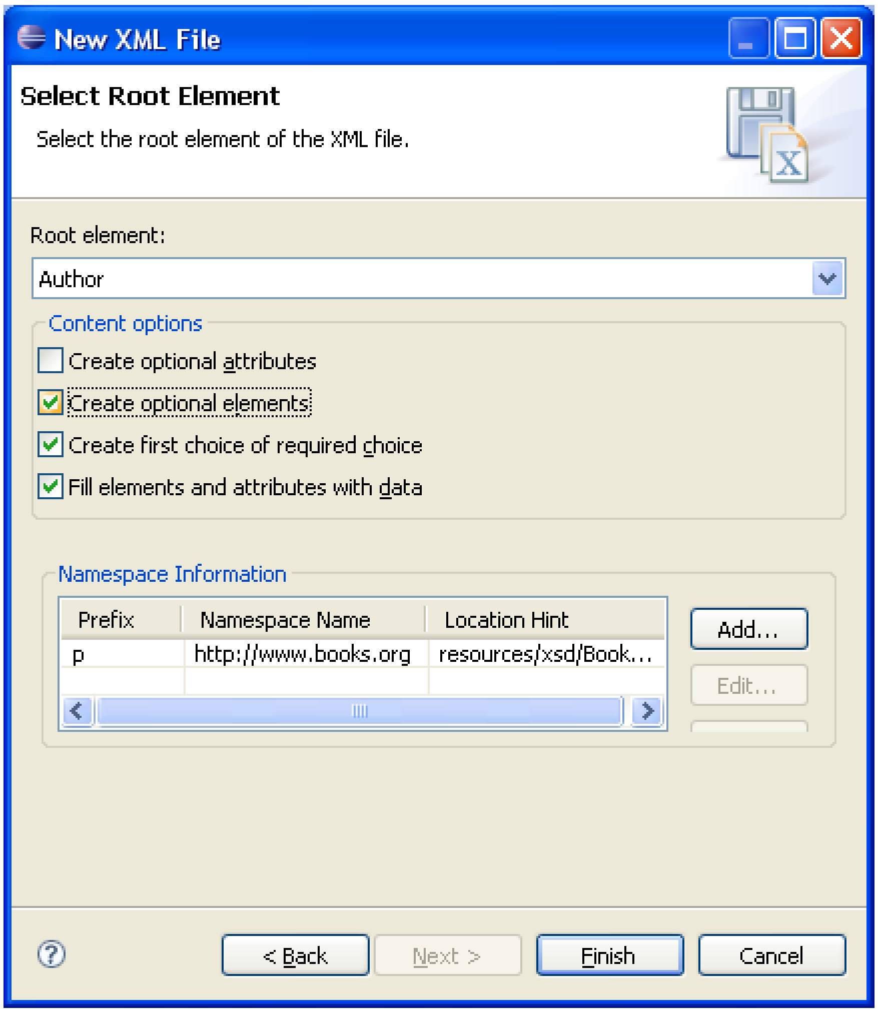


Figure 9.2.3: New XML file dialog box

7. Select the root element and chose appropriate options. Add/edit namespace information, if any and click the **Finish** button.



Select Root Element

Select the root element of the XML file.

Root element: Author

Content options

- [ ] Create optional attributes
- [x] Create optional elements
- [x] Create first choice of required choice
- [x] Fill elements and attributes with data

Namespace Information

Prefix	Namespace Name	Location Hint
p	http://www.books.org	resources/xsd/Book...

Figure 9.2.4: New XML file dialog box

8. Click **Run**. `Output.xml` which contains the result of transformation will be shown.