Getting Started with Kepler

The *Getting Started with Kepler* guide is a tutorial style manual for scientists who want to create and execute scientific workflows.

Table of Contents

1. Introduction

The Getting Started Guide introduces the main components and functionality of Kepler, and contains step-by-step instructions for using, modifying, and creating your own scientific workflows. The Guide provides a brief introduction to the application interface as well as to application-specific terminology and concepts. Once you are familiar with the general principles of Kepler, we recommend that you work through a couple of the sample workflows covered in Section 7 to get a feel for how easy it is to use and modify workflow components and how components can be combined to form powerful workflows.

1.1. What is Kepler?

Kepler is a software application for the analysis and modeling of scientific data. Kepler eliminates most of the programming typically required to create executable models by instead creating visual representations of these processes. These representations, or "scientific workflows," visualize the flow of data among discrete analysis and modeling components (*Figure 1*).

Kepler allows scientists to create their own executable scientific workflows by simply dragging and dropping components onto a workflow creation area and connecting the components to construct a specific data flow, creating a visual model of the analytical portion of their research. Kepler represents the overall workflow visually so that it is easy to understand how data flow from one component to another. The resulting workflow can be saved in a text format, emailed to colleagues, and/or published for sharing with colleagues worldwide.

Kepler is based on distributed computing technologies (see Appendices, Section 8) that allow scientists to share their data and workflows with other scientists and to use data and analytical workflows from others around the world. Kepler also provides access to a continually expanding, geographically distributed set of data repositories, computing resources, and workflow libraries (e.g., ecological data from field stations, specimen data from museum collections, data from the geosciences, etc.).

1.2. What are Scientific Workflows?

Scientific workflows are a flexible tool for accessing scientific data (streaming sensor data, medical and satellite images, simulation output, observational data, etc.) and executing complex analysis on the retrieved data.

Each workflow consists of analytical steps that may involve database access and querying, data analysis and mining, and intensive computations performed on high performance cluster computers. Each workflow step is represented by an “actor,” a seemingly simple component that can be dragged and dropped into a workflow (*Figure 1*) via Kepler’s visual interface. Connected actors (and a few other components that we’ll discuss in later sections) form a workflow, allowing scientists to inspect and visualize data on the fly as it is computed, make parameter changes when necessary, and re-run and reproduce experimental results.[1]

Workflows may represent theoretical models or observational analyses; they can be simple and linear, or complex and non-linear. One of the benefits of scientific workflows is that they are hierarchical, meaning that they can contain sub-workflows to perform tasks (importing data from R or an image processing application, for example). Kepler automates low-level data processing tasks so that scientists can focus instead on the scientific questions of interest.

Scientific workflows provide access to the benefits of today’s grid technologies (providing access to distributed resources such as data and computational services), while hiding the underlying complexity of those technologies. Workflows also provide:

- documentation of all aspects of an analysis;
- visual representation of analytical steps;
- ease of testing and debugging;
- scalability;
- reproducibility of a given project with little effort; and
- reuse of part or all of a workflow in a different project.

To date, most scientific workflows have involved a variety of software programs and sophisticated programming languages. Kepler builds upon the open-source Ptolemy II visual modeling system (http://ptolemy.eecs.berkeley.edu/ptolemyII/), creating a single work environment for scientists. The result is a user-friendly program that allows scientists to create their own scientific workflows without having to integrate several different software programs or enlist the assistance of computer programmers.

A number of ready-to-use, components come standard with Kepler, including generic mathematical, statistical, and signal processing components and components for data input, manipulation, and display. R- or Matlab-based statistical, image processing, and GIS functionality are available through direct links to these external packages. You may also create new components or wrap existing components from other programs (e.g., C programs) for use within Kepler. For more information about the design of Kepler and Ptolemy, upon which Kepler is based, see the Appendices.

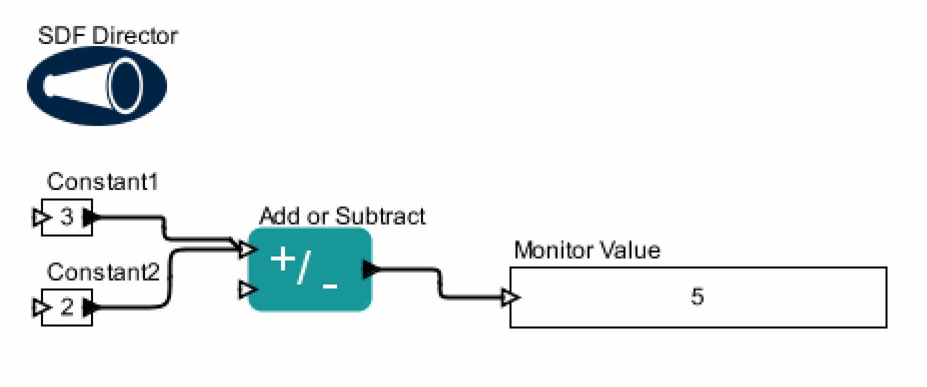


Figure 1: A simple scientific workflow developed in Kepler

Figure 1 illustrates a simple workflow that adds two numbers and displays the result. The two components (or "actors") labeled *Constant1* and *Constant2* produce constant values that flow to the *Add or Subtract* actor, which subsequently processes these values. The *Add or Subtract* actor calculates the sum of the input values, and then produces that sum. The *Monitor Value* actor opens a small window and displays the sum, in this case the value 5. The *SDF Director* controls the workflow, specifying when each actor should perform.

[1] See Ludäscher, B., I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, Y. Zhao. 2005. Scientific Workflow Management and the Kepler System http://users.sdsc.edu/~ludaesch/Paper/kepler-swf.pdf

2. Basic Components in Kepler

Scientific workflows consist of customizable components—directors, actors, and parameters—as well as relations and ports, which facilitate communication between the components.

2.1. Director and Actors

Kepler uses a director/actor metaphor to visually represent the various components of a workflow. A director controls (or directs) the execution of a workflow. The actors take their execution instructions from the director. In other words, actors specify *what* processing occurs while the director specifies *when* it occurs.

Every workflow must have a director that controls the execution of the workflow using a particular model of computation. Each model of computation in Kepler is represented by its own director. For example, workflow execution can be synchronous, with processing occurring one component at a time in a designated sequence (*SDF Director*). Alternatively, workflow components can execute in parallel, with one or more components running simultaneously (which might be the case with a *PN Director*). A small set of commonly used directors come pre-packaged with Kepler, but more are available in the underlying Ptolemy II software that can be accessed as needed. For more detailed discussion of workflow models of computation, see the Appendices and refer to Ptolemy II documentation.

Kepler identifies two kinds of actors: atomic and composite. Atomic actors represent a single data source or an operation. Composite actors are collections or sets of actors bundled together to perform more complex operations within a hierarchical, nested workflow. In essence, an entire workflow can be represented as a composite actor and included as a component within a higher-level workflow. In more complex hierarchical workflows, it is possible to have different directors at different levels.

Kepler provides a large set of packaged actors for creating and editing scientific workflows. Actors can be added to Kepler for an individual's exclusive use and/or can be made available to others.

2.2. Ports

Each actor in a workflow can contain one or more ports used to consume or produce data and communicate with other actors in the workflow. Actors are connected in a workflow via their ports. The link that represents data flow between one actor port and another actor port is called a channel. Ports are categorized into three types:

- input port – for data consumed by the actor;
- output port – for data produced by the actor; and
- input/output port – for data both consumed and produced by the actor.

Each port is configured to be either a "singular" or "multiple" port. A single input port consumes only a single value and can be connected to only a single channel, whereas a multiple input port can take several inputs and therefore can be connected to multiple channels. Single ports are designated with a dark triangle; multiple ports use a hollow triangle.

Workflows can also use external ports and port parameters. See the Ptolemy documentation for more information.

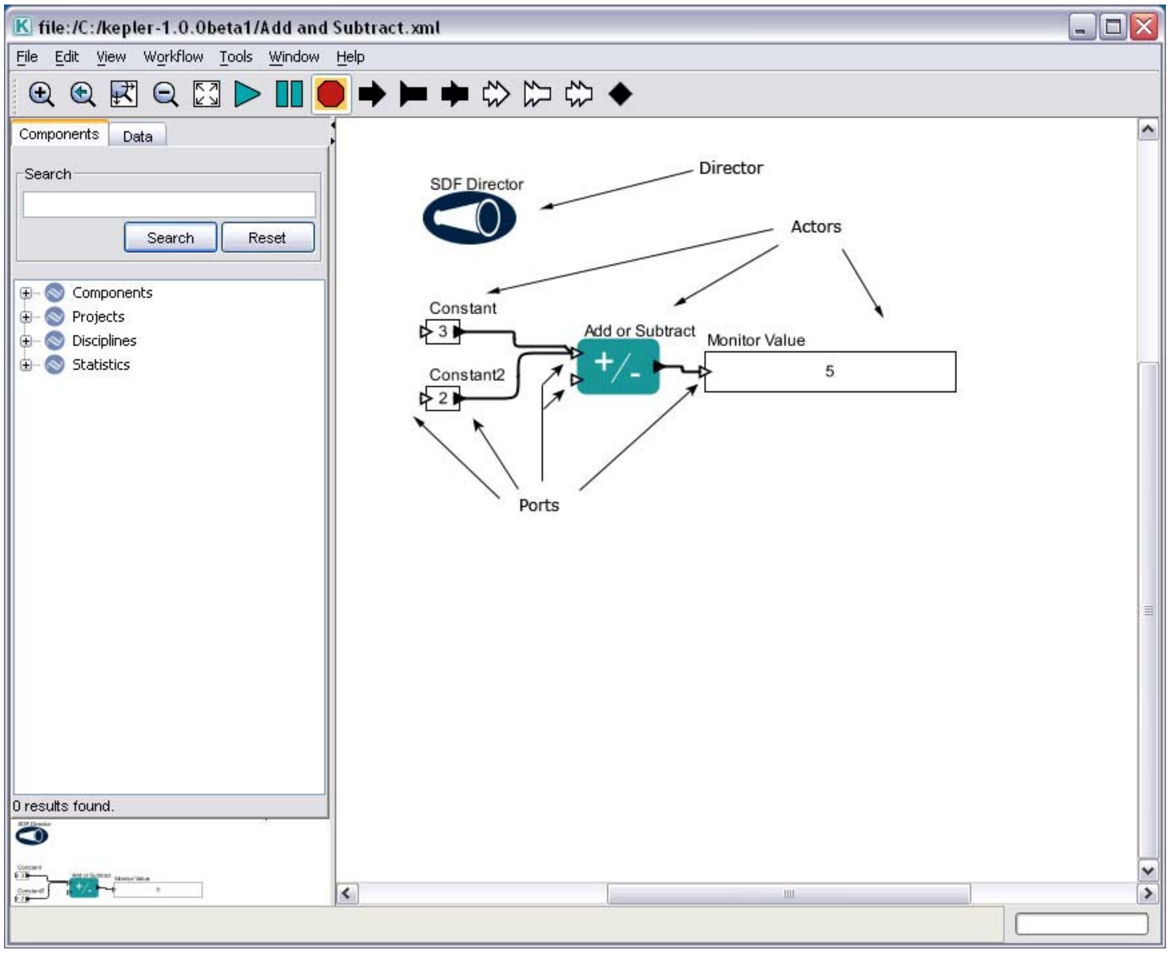


Figure 2: Main window of Kepler with some of the major workflow components highlighted

2.3. Relations

Relations are used to direct the same input or output to more than one other port. Using a relation, a singular port can output data to several ports, just like a multiple port. For example, a scientist might wish to direct the output of an operational actor to another operational actor for further processing, but also to a display actor to visualize the data at that specific reference point. For an example of a Relation, see Section 7.3.

2.4. Parameters

Parameters are configurable values that can be attached to a workflow or to individual directors or actors. For example, the *Scatterplot* actor has parameters that allow the user to set the scale of the graph on the X and Y axis. The parameters of simulation model actors can be configured to control certain aspects of the simulation, such as initial values. Director parameters control the number of workflow iterations and the relevant criteria for each iteration. For more information about configuring parameters, see Section 6.2.2

The next sections include information on downloading and installing Kepler, and then provide an overview of the interface and step-by-step examples of how to open, edit, and run different scientific workflows.

3. Downloading and Installing Kepler

Kepler is an open-source, cross-platform software program that can run on Windows, Macintosh, or Linux-based platforms. Kepler can be downloaded from the project website: http://kepler-project.org.

Although Kepler has been designed to be a stable and full-featured application, the software is still a research prototype, and Kepler releases are a continual work in progress. Kepler users are encouraged to suggest improvements for the documentation, new features, new actors and components, etc., as well as notify the designers of bugs and other problems. See http://www.kepler-project.org/Wiki.jsp?page=GettingInvolved for more information. Community involvement in the on-going development of Kepler has proved valuable because it allows the system to quickly adapt to the needs of practicing scientists. To stay abreast of changes and updates, subscribe to the Kepler users mailing list at http://mercury.nceas.ucsb.edu/ecoinformatics/mailman/listinfo/kepler-users.

3.1. System Requirements

To run Kepler your system will need at least:

- 270 MB (283,688,960 bytes) of disk space
- 128 MB of RAM
- Java 1.4 or higher*

To download and install Kepler, follow the instructions for your system. Downloading the installer files may be time consuming depending upon your connection.

<u>*NOTE:</u> Java 1.4 or higher is required and can be obtained from Sun's Java website at: http://java.sun.com/j2se/downloads/ or from your system administrator. Some Kepler installations include Java 1.4 and others do not. Check to see if your version of Kepler includes Java before downloading it from http://kepler-project.org/Wiki.jsp?page=Downloads

3.2. Installing on Windows

Two versions of Kepler are available for installation on a Windows machine: one with Java; and one without Java. If you do not have Java 1.4 (or higher) installed, be sure to download the Kepler package that includes Java or download and install Java from Sun's website. Follow these steps to download and install Kepler for Windows:

1. Click the following link: http://kepler-project.org/Wiki.jsp?page=Downloads and select the desired Windows version (with or without Java).
2. Save the install file to your computer.
3. Double-click the install file to open the install wizard.
4. Follow the steps presented to complete the Kepler installation process.

Once the installation process is complete, a Kepler application icon will appear on your desktop (*Figure 3*).

3.3. Installing on Macintosh

Java is included in all versions of Kepler for Mac OSX and thus can be assumed to be present. Follow these steps to download and install Kepler for Macintosh systems:

1 Click the following link: http://kepler-project.org/Wiki.jsp?page=Downloads and select the Mac install file. Save the zipped install file to your computer. Once the file has been saved to your computer, the zipped install file should automatically begin to extract itself. (If the extraction does not start automatically, manually extract the zip file by double-clicking it.)
2 Double-click the install icon that appears on your desktop once the extraction is complete.
3 Follow the steps presented in the install wizard to complete the Kepler installation process.

Once the installation process is complete, a Kepler application icon will appear on your desktop (*Figure 3*).

Figure 3: Kepler shortcut

3.4. Installing on Linux

After making sure you have Java 1.4 or higher installed, follow these steps to download and install Kepler on Linux:

1. Click the following link: http://kepler-project.org/Wiki.jsp?page=Downloads and select the Linux install file.
2. Save the zipped install file to your computer.
3. Open a shell window and extract the zipped install file to the desired directory.

Once the Kepler file is unzipped, the installation is complete.

4. Starting Kepler

To start Kepler, follow the instructions for your platform.

4.1.1. Windows and Macintosh Platforms

Double-click the Kepler icon on the desktop (*Figure 3*).

The main Kepler application window opens *(Figure 4).* From this window you can access and run sample and existing scientific workflows and/or create your own custom scientific workflow. Each time you open an existing workflow or create a new workflow, a new application window will open. Multiple windows allow you to work on several workflows simultaneously and compare, copy, and paste components between workflows

4.1.2. Linux Platform

1. Open a shell window.
2. Navigate to the directory into which you installed Kepler.
3. Type sh ./kepler.sh

The main Kepler application window opens *(Figure 4).* From this window you can access and run sample and existing scientific workflows and/or create your own custom scientific workflow. Each time you open an existing workflow or create a new workflow, a new application window will open. Multiple windows allow you to work on several workflows simultaneously and compare, copy, and paste components between workflows.

5. Kepler Interface

Scientific workflows are edited and built in Kepler's easily navigated, drag-and-drop interface. The major sections of the Kepler application window are:

- Menu bar – provides access to all Kepler functions.
- Toolbar – provides access to the most commonly used Kepler functions.
- Components and Data Access area – consists of a Components tab and Data tab. Both tabs contain a search function and display the library of available components and/or search results.
- Workflow canvas – provides space for displaying and creating workflows.
- Navigation area – displays a portion of a workflow if it is too big to fit into the Workflow canvas.

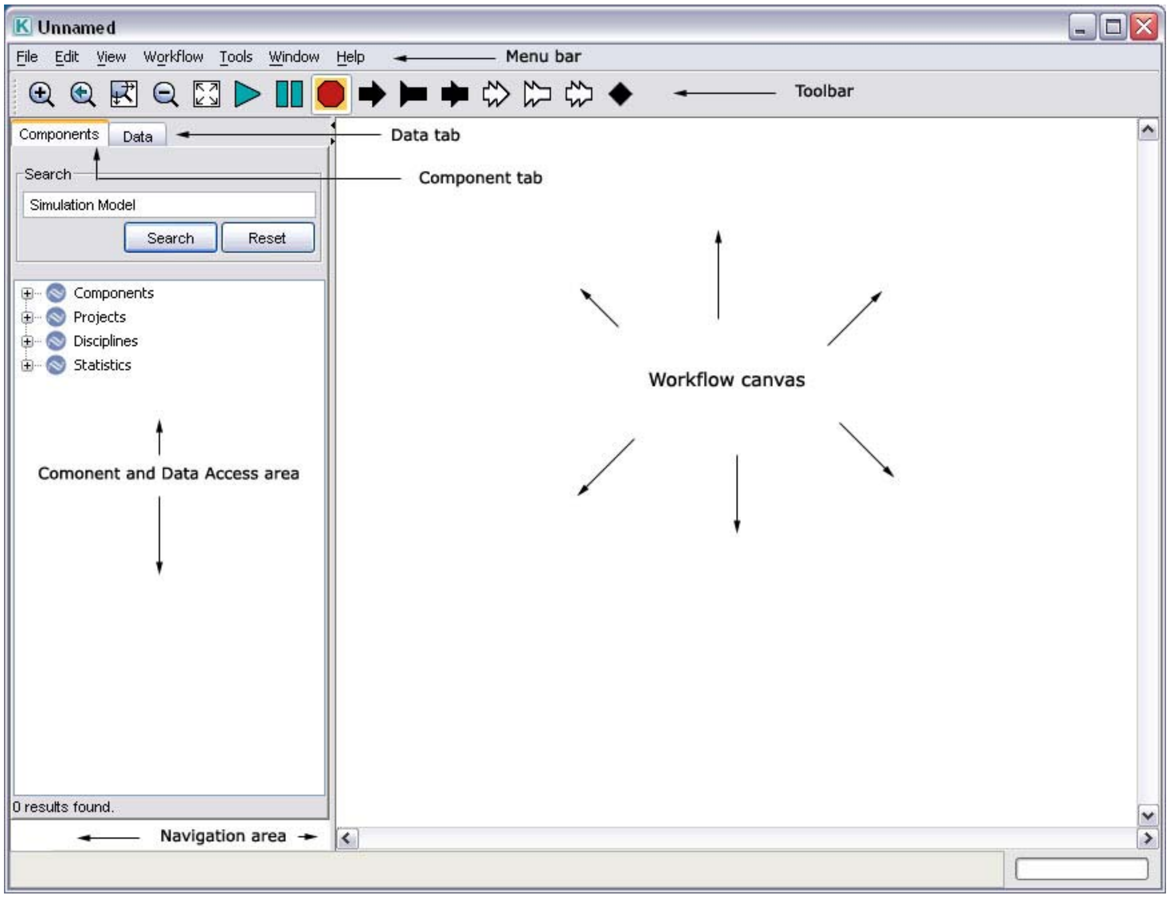


Figure 4: Empty Kepler window with major sections annotated

5.1. The Toolbar

The Kepler toolbar is designed to contain the most commonly used Kepler functions (*Figure 5*).

The main sections of the toolbar include:

- Viewing –zoom in, reset, fit, and zoom out of the model on the Workflow canvas
- Running – run, pause, and stop the model without opening the Run window
- Ports – add single (black) or multi (white) input and output ports to workflows; add Relations to workflows

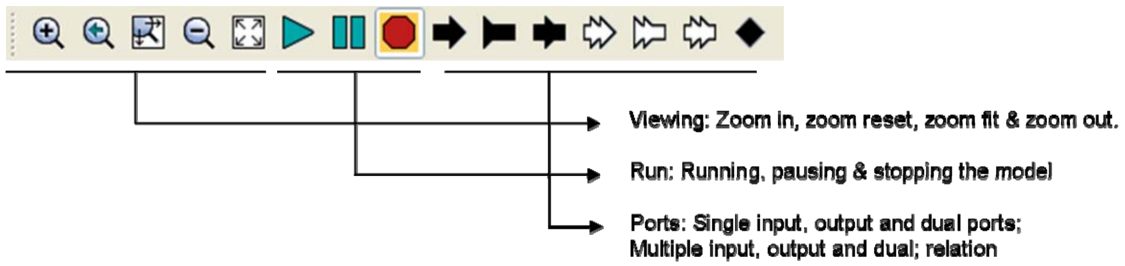

Viewing: Zoom in, zoom reset, zoom fit & zoom out.
Run: Running, pausing & stopping the model
Ports: Single input, output and dual ports;
Multiple input, output and dual; relation

Figure 5: Annotated Kepler Toolbar

5.2. Components and Data Access Area

The Components and Data Access area contains a library of workflow components (under the Components tab) and a search mechanism for locating and using data sets (under the Data tab). When you first open the application, the Components tab is displayed.

Components in Kepler are arranged in ontologies. You can browse for components by clicking through the ontology trees, or use the search function at the top of the Components tab to find a specific component.

Four component ontologies appear in the Component tab. Each ontology categorizes components in a unique way. Components may be listed within multiple categories, and searching for a component may return results from several locations. You may use either instance of the actor—only its categorization is different. For more information about searching for components, see section 6.4.2.

Ontology	**Description**
Components	Contains a standard library of components, arranged by function.
Projects	Contains a library of project-specific components (e.g., SEEK or CIPRes)
Disciplines	Contains a library of components arranged by discipline (e.g., Chemistry or Ecology)
Statistics	Contains a library of components for use with statistical analysis.

Table 1: Component Ontologies in Kepler

Click the Data tab to reveal the Data Access area. From here, you can easily search the EcoGrid for remotely hosted data sets. For more information about searching for data, see section 6.4.1.

5.3. Director and Actor Icons

In Kepler, a single class of icons represents directors. Actors are divided into functional categories, with each category assigned to visually related icons as listed below. For a complete list of actors, including their specific icons and a description of each actor, see Appendix 8.3, "Actor Reference."

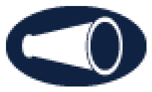
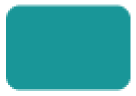
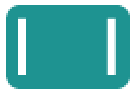
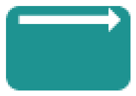
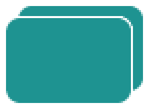
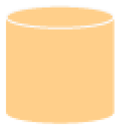
Icon	Name	Description
	Director	Stand-alone component that directs the other components (the actors) in their execution
	Atomic actor	A connected component that can be a single data source or an operation upon data
	Display actor	A connected component that outputs the workflow in text or graphical format
	Model actor	A connected component that creates and executes an analytic rule set
	Composite actor	Collections or sets of actors bundled together to perform more complex operations within a hierarchical, nested workflow
	Data	Data sets located from the EcoGrid or local computer containing either data or metadata or both.

Table 2: The major Kepler icons

5.4. The Workflow Canvas

Scientific workflows are opened, created, and modified on the Workflow canvas. Components are easily dragged and dropped from the Component and Data Access area to the desired canvas location. Each component is represented by an icon (see Section 5.3 for examples), which makes identifying the components simple. Connections between the components (i.e., channels) are also represented visually so that the flow of data and processing is clear.

Each time you open an existing workflow or create a new workflow, a new application window will open. Multiple windows allow you to work on several workflows simultaneously and compare, copy, and paste components between Workflow canvases.

6. Basic Operations in Kepler

This section will cover the basic operations in Kepler: opening and running an existing workflow, and some of the basic functions needed to edit, design, and create your own workflows.

6.1. Opening an Existing Scientific Workflow

This section will walk you through the steps of opening a workflow in Kepler. We begin the generic steps and then specify the steps to open a pre-existing workflow in Kepler. To open any existing workflow:

1. From the Menu bar, select File, then Open File. A standard file dialog box will appear.
2. If the file dialog box does not open to the "/kepler" directory (the directory name will reflect the application version, e.g., kepler-1.0.0beta1), then navigate to the "/kepler" directory, then to /demos/getting-started.
3. Double-click a workflow file to open it. The workflow will appear in the Workflow canvas of the application window.

NOTE: In order to run properly, some workflows require online access (to access remote data, for example).

6.1.1. Example 1: Opening the Lotka-Volterra Workflow

In this example we will open a specific workflow: the classic predator pray model, the Lotka-Volterra workflow. To open this workflow:

1. From the Menu bar, select File, then Open File. A standard file dialog box will appear (*Figure 6*).
2. Navigate to the "/kepler/demos/getting-started/" directory and locate the file named "02-LotkaVolterraPredatorPrey.xml".

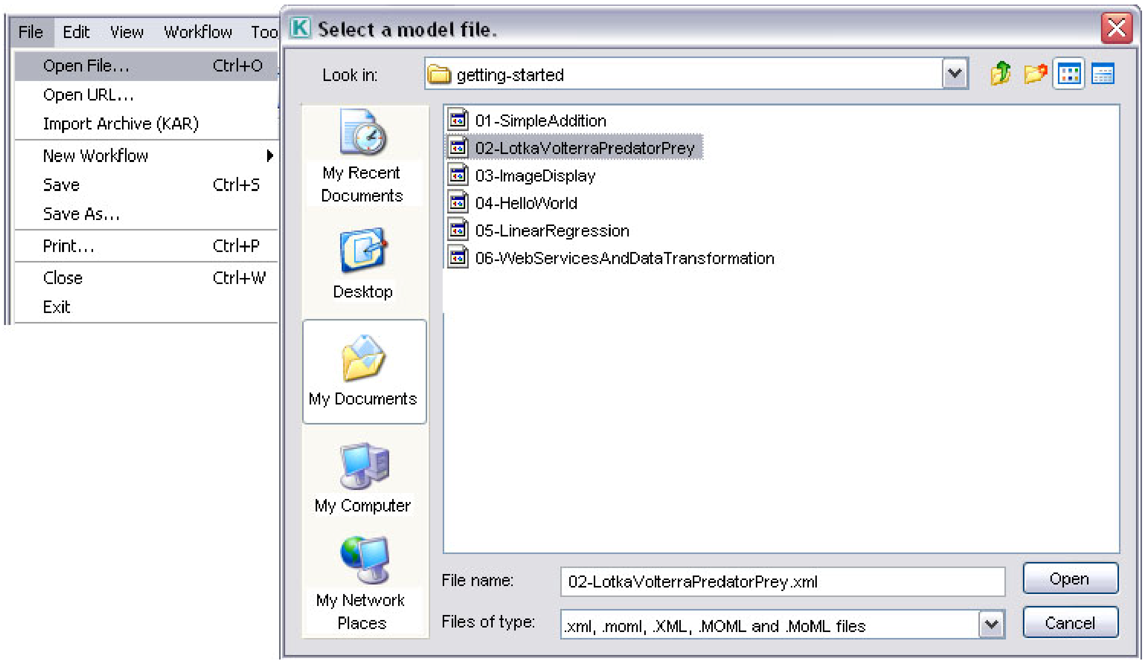


Figure 6: Graphic showing navigation to Lotka-Volterra workflow. The workflow is in the "/kepler/demos/getting-started/" directory.

3. Double-click the "02-LotkaVolterraPredatorPrey.xml" file. The Lotka-Volterra workflow appears in the Workflow canvas of the application window (*Figure 7*).

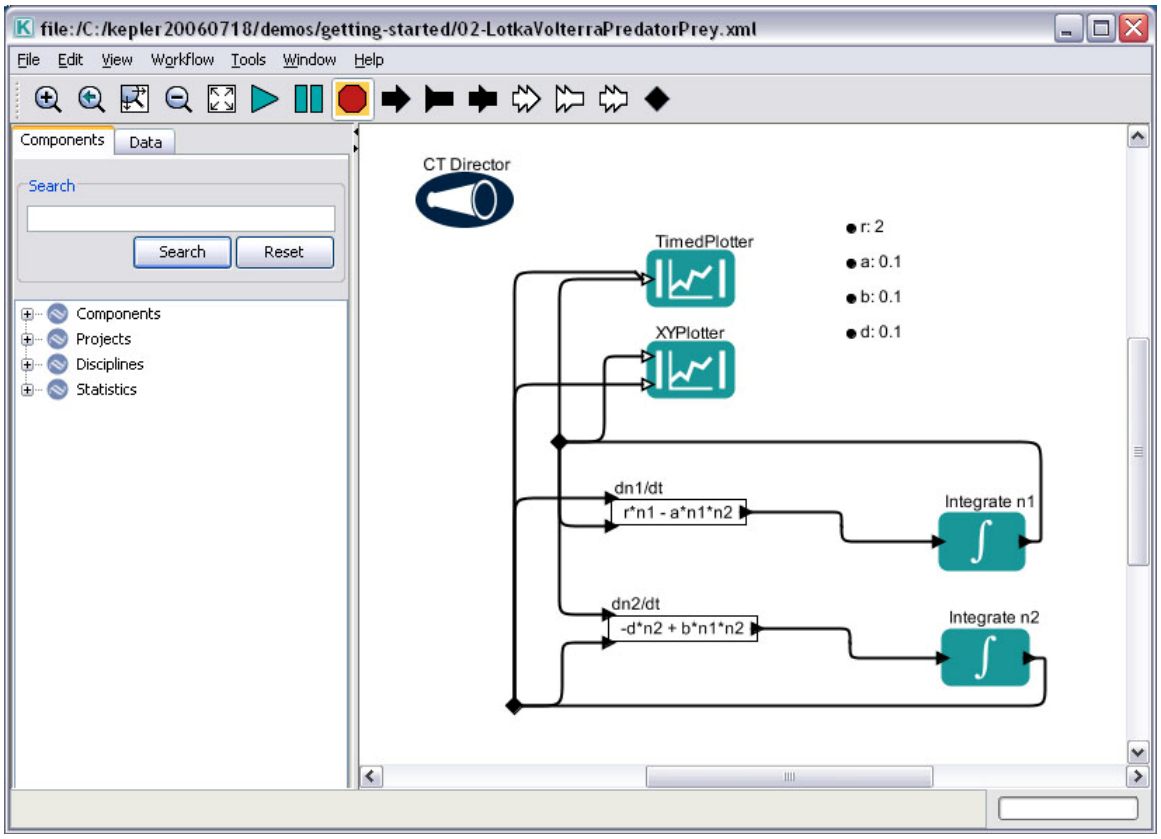


Figure 7: The Lotka-Volterra workflow in the Kepler interface.

6.2. Running an Existing Scientific Workflow

This section will walk through the steps of running an existing workflow.

To run any existing scientific workflow:

1. Open the desired workflow.
2. From the Toolbar, select the Run button.
3. The workflow will execute and produce the specified output.

OR

1. Open the desired workflow.
2. From the Menu bar, select Workflow, then Runtime Window. A Run window will appear. If the workflow has parameters, they will appear here.
3. Adjust the parameters as needed, and then click the Go button.
4. The workflow will execute and produce the specified output. During workflow execution, you may select the Pause, Resume, or Stop buttons.

<u>NOTE</u>: The ability to pause, resume, or stop the workflow execution is not available when executing a workflow via the Run button. Additionally, the option to adjust workflow parameters, if they exist, is not available via the Run button. The workflow will execute with the default parameter values.

6.2.1. Example 2: Running the Lotka-Volterra Workflow with Default Parameters

The Lotka-Volterra model uses the continuous time domain in Kepler to solve two coupled differential equations: one that models the predator population; and one that models the prey population. The results are plotted as they are calculated, showing both populations change and a phase diagram of the dynamics. For more information about the model, see Section 6.2.2. To run the Lotka-Volterra workflow:

1. Open the workflow file named “02-LotkaVolterraPredatorPrey” from the “/kepler/demos/getting-started/” directory.
2. From the Menu bar, select Run.
3. The Lotka-Volterra workflow will execute with the default parameters and produce two graphs. The graph labeled TimedPlotter depicts the interaction of predator and prey over time (i.e., the cyclical changes of the predator and prey populations over time predicted by the model). The graph labeled XYPlotter depicts a phase portrait or the population cycle around the equilibrium (i.e., the predator population against the prey population). Together these graphs show how the predator and prey populations are linked: as prey increases, the number of predators increase. (*Figure 8*)

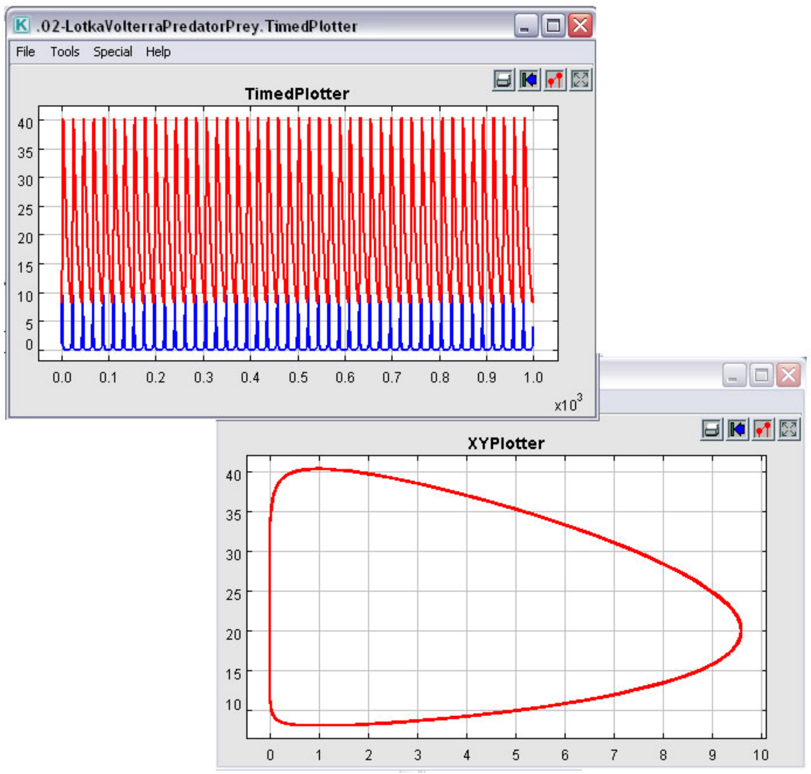


Figure 8: Graphics from running the Lotka-Volterra workflow

6.2.2. Example 3: Running the Lotka-Volterra Workflow with Adjusted Parameters

To better illustrate the effect of parameters on a workflow, we must first provide some background about the Lotka-Volterra workflow.

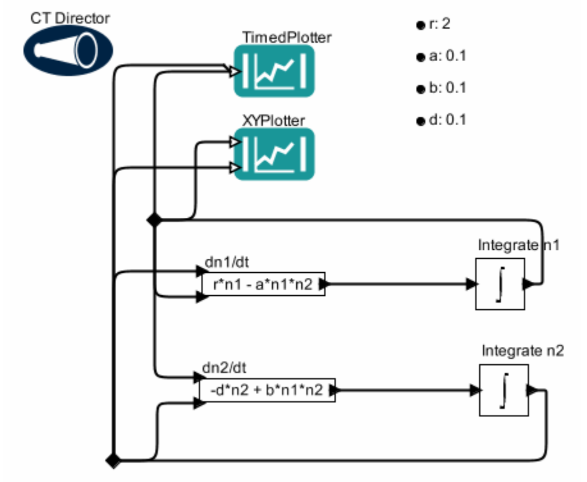


Figure 9: Graphic of Lotka-Volterra workflow

The Lotka-Volterra model was developed independently by Lotka (1925) and Volterra (1926) and is made up of two differential equations. One describes how the prey population changes (dn1/dt = r*n1 - a*n1*n2), and the second equation describes how the predator population changes (dn2/dt = -d*n2 + b*n1*n2).

The Lotka-Volterra model is based on certain assumptions:

- the prey has unlimited resources;
- the prey's only threat is the predator;
- the predator is a specialist (i.e., the predator's only food supply is the prey); and
- the predator's growth depends on the prey it catches.

The Lotka-Volterra model as represented in Kepler as a scientific workflow contains:

- six actors - two plotters, two equations, and two integral functions;
- one director; and
- four workflow parameters.

NOTE: The director has several configurable parameters as do the two plotter actors.

The critical assumptions above provide the basis for the workflow parameters. The workflow parameters and their defaults are as follows:

Parameter	**Default Value**	**Description**
r	2	the intrinsic rate of growth of prey in the absence of predation
a	0.1	capture efficiency of a predator or death rate of prey due to predation
b	0.1	proportion of consumed prey biomass converted into predator biomass (efficiency of turning prey into new predators)
d	0.1	death rate of the predator

Table 3: Description of the default parameters for the Lotka-Volterra workflow

In the differential equations used in the workflow, the variable n1 represents prey density, and the variable n2 represents predator density.

When changing parameters in a workflow, the assumptions of the model must be kept in mind. For example, if creating a Lotka-Volterra model with rabbits as prey and foxes as predators, the following assumptions can be made with regard to how the rabbit population changes in response to fox population behavior:

- the rabbit population grows exponentially unless it is controlled by a predator;
- rabbit mortality is determined by fox predation;
- foxes eat rabbits at a rate proportional to the number of encounters;

- the fox population growth rate is determined by the number of rabbits they eat and their efficiency of converting the eaten rabbits into new baby foxes; and
- fox mortality is determined by natural processes.

If you think of each run of the model in terms of rates at which these processes would occur, then you can think of changing the parameters in terms of percent of change over time.

To run the Lotka-Volterra workflow with adjusted parameters:

1. Open the workflow file named “02-LotkaVolterraPredatorPrey” from the “/kepler/demos/getting-started/” directory
2. From the Menu bar, select Workflow, then Runtime Window. The Runtime window will appear. Notice there are two sets of parameters – one for the workflow and one for the director. In this example, you will make adjustments to both sets of parameters.

3. Adjust the workflow parameters as suggested in Table 4.

Parameter	**Value**	**Description**
r	4	the intrinsic rate of growth of prey in the absence of predation
a	0.05	capture efficiency of a predator or death rate of prey due to predation
b	0.03	proportion of consumed prey biomass converted into predator biomass (efficiency of turning prey into new predators)
d	0.04	death rate of the predator

Table 4: Description of the suggested parameters for the Lotka-Volterra workflow

4. Adjust the value of the `stopTime` director parameter to 100.
5. In the Runtime window, click the Go button.

The Lotka-Volterra workflow will execute with the adjusted parameters and produce the two graphs: 1) the TimedPlotter graph and 2) the XYPlotter graph. Note that with the changes in the parameters, the relationship between the predator and prey populations are still linked but the relationship has changed.

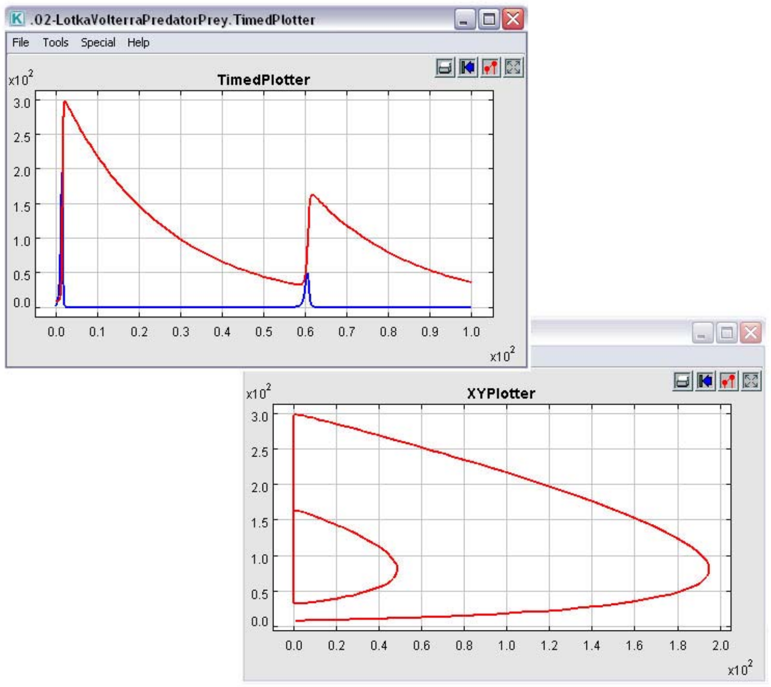


Figure 10: Graphs from running the Lotka-Volterra model with adjusted parameters

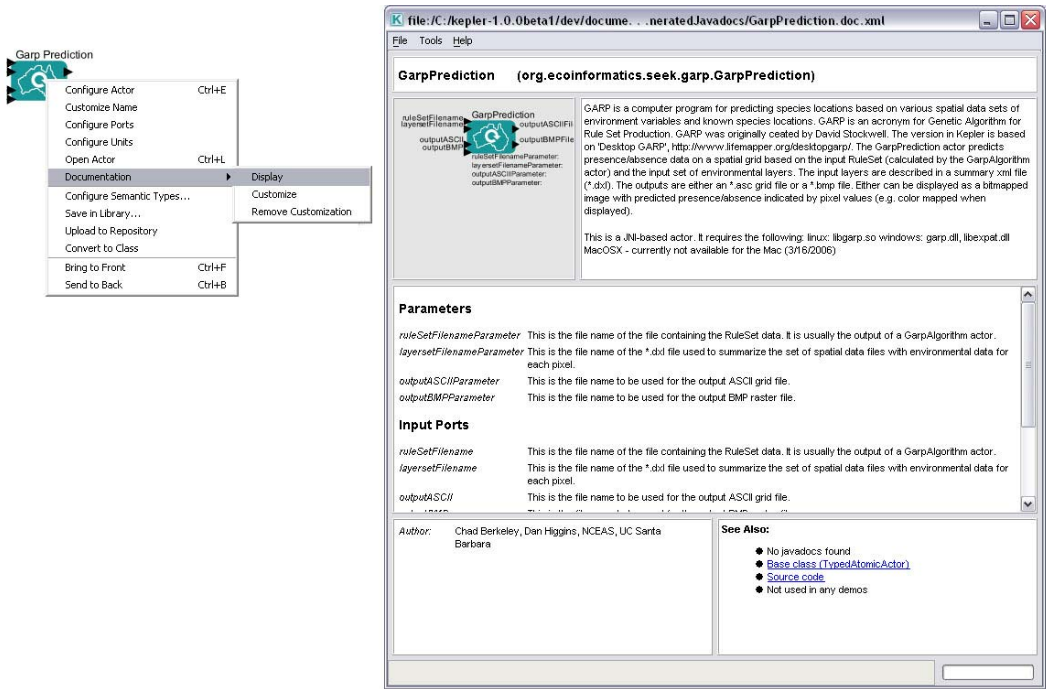
6.3. Editing an Existing Scientific Workflow

There are two ways to edit an existing scientific workflow:

- substitute a different data set for the current data set; or
- substitute one or more analytical processes in the workflow with other analytical processes (e.g., substitute a neural network model actor for a probabilistic model actor).

Before substituting data or processes, you must understand the required inputs and outputs of the actors involved.

NOTE: **To see a high-level description of an actor, right-click that actor to display a menu; select Documentation, then Display (*Figure 11*). A dialog box containing a description of the main function of the actor and its required inputs and output appears. When finished with this dialog, close the window.**



GarpPrediction (org.ecoinformatics.seek.garp.GarpPrediction)

GARP is a computer program for predicting species locations based on various spatial data sets of environment variables and known species locations. GARP is an acronym for Genetic Algorithm for Rule Set Production. GARP was originally ceated by David Stockwell. The version in Kepler is based on 'Desktop GARP', http://www.lifemapper.org/desktopgarp/. The GarpPrediction actor predicts presence/absence data on a spatial grid based on the input RuleSet (calculated by the GarpAlgorithm actor) and the input set of environmental layers. The input layers are described in a summary xml file (*.dxl). The outputs are either an *.asc grid file or a *.bmp file. Either can be displayed as a bitmapped image with predicted presence/absence indicated by pixel values (e.g. color mapped when displayed).

This is a JNI-based actor. It requires the following: linux: libgarp.so windows: garp.dll, libexpat.dll MacOSX - currently not available for the Mac (3/16/2006)

Parameters

ruleSetFilenameParameter	This is the file name of the file containing the RuleSet data. It is usually the output of a GarpAlgorithm actor.
layersetFilenameParameter	This is the file name of the *.dxl file used to summarize the set of spatial data files with environmental data for each pixel.
outputASCIIParameter	This is the file name to be used for the output ASCII grid file.
outputBMPParameter	This is the file name to be used for the output BMP raster file.

Input Ports

ruleSetFilename	This is the file name of the file containing the RuleSet data. It is usually the output of a GarpAlgorithm actor.
layersetFilename	This is the file name of the *.dxl file used to summarize the set of spatial data files with environmental data for each pixel.
outputASCII	This is the file name to be used for the output ASCII grid file.

Author: Chad Berkeley, Dan Higgins, NCEAS, UC Santa Barbara

See Also:

- No javadocs found
- Base class (TypedAtomicActor)
- Source code
- Not used in any demos

Figure 11: Displaying actor documentation

To edit an existing scientific workflow:

1. Open the desired workflow.
2. Identify which workflow component is the target for substitution.
3. Select the target component (data actor or processing actor) by clicking it. The selected component will be highlighted in a thick yellow border.
4. Press the Delete key on your keyboard. The highlighted component will disappear from the Workflow canvas.
5. From the Components and Data Access area, drag either an appropriate data file or processing actor to the Workflow canvas.
6. Connect the appropriate input and output ports.
7. Run the workflow.
8. From the menu bar, select File, then Save (to save over the existing workflow) or Save As (to save as a new workflow). If using the Save As option, enter a new workflow name when prompted.

6.3.1. Example 4: Editing/Substituting Analytical Processes in the Image J Workflow

In this example, we will show how two different actors can perform the same function in a workflow. We will work with the Image Display workflow found under "/kepler/demos/getting-started/", and we will substitute the *Browser* actor for the *ImageJ* actor. Both actors will display a GARP (Genetic Algorithm for Rule-set Production) image of a species distribution of the species Mephitis throughout North and South America. GARP is a genetic algorithm that creates an ecological niche model for a species that represents the environmental conditions where that species would be able to maintain populations. GARP was originally developed by David Stockwell, at the San Diego Supercomputer Center. For more information on GARP, see http://www.lifemapper.org/desktopgarp/.

To edit the Image Display workflow:

1. Open the 03-Image-Display.xml workflow from the "/kepler/demos/getting-started/" directory.
2. Select the target component, the *ImageJ* actor in this case. The *ImageJ* actor will be highlighted in a thick yellow border, indicating that it is selected (*Figure 12*).

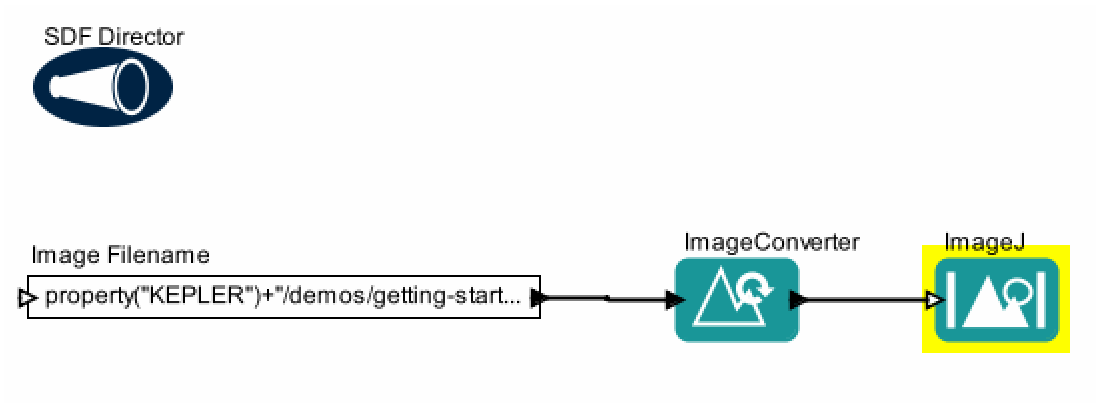


Figure 12: Image Display workflow showing *ImageJ* actor highlighted

3. Press the Delete key on your keyboard. The *ImageJ* actor will disappear from the Workflow canvas.
4. From the Components and Data Access area, drag the *BrowserDisplay* actor to the Workflow canvas. You can find the *BrowserDisplay* actor in the Components tab under "Components > Data Output > Workflow Output > Textual Output."
5. Connect the output port of the *ImageConverter* actor to the input port of the *BrowserDisplay* actor. To connect the ports, left-click and hold on the output port (black triangle) on the right side of the *ImageConverter* actor, drag the pointer to the upper input port on the left side of the *BrowserDisplay* actor, and then release the mouse. If the connection is made, you'll see a thick black line. If the connection is not completely made, the line will be thin.
6. Run the workflow.
7. From the Menu bar, select File, then Save (to save over the existing workflow) or Save As (to save as a new workflow). If using the Save As option, enter a new workflow name when prompted.

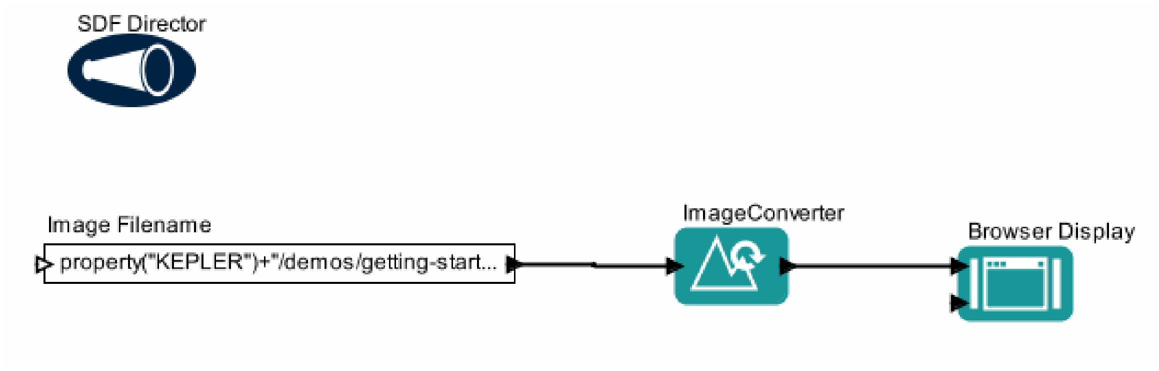


Figure 13: The Image Display workflow with the *Browser Display* actor substituted for the *ImageJ* actor.

NOTE: Sometimes the easiest way to connect actors is to go from the output port to the input port.

6.4. Searching in Kepler

Kepler provides a searching mechanism to locate data (on the Grid) and analytical processing components (on the local system). The examples given in this section describe searching for data and components in Kepler.

6.4.1. Searching for Available Data

Via its search capabilities, Kepler provides access to data from the EcoGrid. EcoGrid resources are stored in the KNB Metacat http://knb.ecoinformatics.org, the KU Digir http://www.specifysoftware.org/Informatics/informaticsdigir/, and the GEON http://www.geongrid.org/ databases. To search for data on the EcoGrid through Kepler:

1. In the Components and Data Access area, select the Data tab (*Figure 14*).
2. Type in the desired search string (e.g., Datos Meteorologicos). Make sure that the search string is spelled correctly.
3. Click the Search button. The search may take several moments. When the search is complete, a list of search results (i.e., Data actors) will be displayed in the Components and Data Access area.
4. To use one or more data actors in a workflow, simply drag the desired actors to the Workflow canvas.

Information about a Data actor can be revealed in two ways: on the Workflow canvas, roll over the Data actor's data output ports to reveal a tool tip containing the name and type of data; Right-click the Data actor and select Get Metadata to open a window containing more information about the data set.

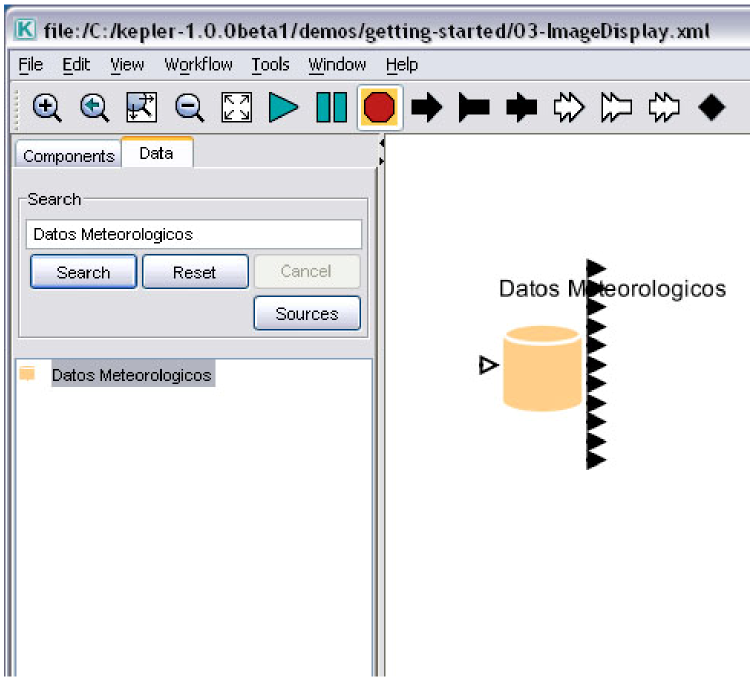

file:/C:/kepler-1.0.0beta1/demos/getting-started/03-ImageDisplay.xml
File Edit View Workflow Tools Window Help
Components
Data
Search
Datos Meteorologicos
Search
Reset
Cancel
Sources
Datos Meteorologicos
Datos Meteorologicos

Figure 14: Searching for and locating Datos Meteorologicos

NOTE: To configure the search, click the Sources button. Select the sources to be searched and the type of documents to be retrieved.

6.4.2. Searching for Available Processing Components

Kepler comes standard with over 200 workflow components and the ability to modify and create your own. You can create an innumerable number of workflows with a variety of analytic functions. The default set of Kepler processing components is displayed under the Components tab in the Components and Data Access area. Components are organized by function (e.g., “Director” or “Filter Actor”). To search for processing components:

1. In the Components and Data Access area to the left of the Workflow canvas, select the Components tab.
2. Type in the desired search string (e.g., “File Fetcher”).
3. Click the Search button. When the search is complete, the search results are displayed in the Components and Data Access area. The search results replace the default list of components. You may notice multiple instances of the same component (Because components are arranged in ontologies, the same component may appear in multiple places in the search results.)
4. To use one or more processing components in a workflow, simply drag the desired components to the Workflow canvas.
5. To clear the search results and re-display the list of default components, click the Reset button.

<u>NOTE:</u> If you know which component you want to use and its location in the Component library, you can navigate to it directly, and then drag it to the Workflow canvas.

6.5. Creating a Basic Scientific Workflow

One of the strengths of Kepler is the ability to design, create, and save your own executable workflows. The general steps in creating a workflow are as follows:

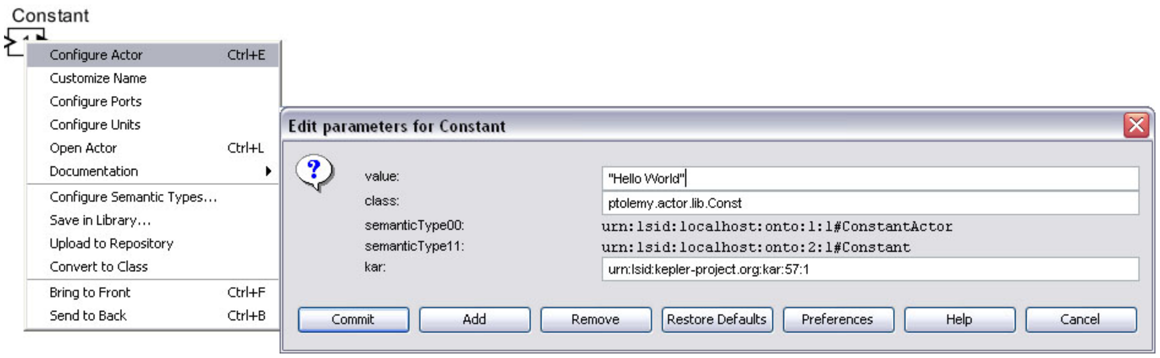
1. Create a conceptual (paper or other medium) model of your scientific workflow.
2. Open the Kepler application.
3. Map the data and actor components available in Kepler to your conceptual model.
4. Select a director for your workflow and drag it to the Workflow canvas.
5. Drag the desired workflow components to the Workflow canvas.
6. Connect the workflow components.
7. Save the workflow.

The examples in this section illustrate how to begin to create your own workflows. The first example is the classic "Hello World" workflow that demonstrates how easy it is to create a functioning workflow in Kepler. The second example is more practical and shows how to use your desktop data in a workflow.

6.5.1. Example 1: Creating a "Hello World" Workflow

To create the "Hello World" workflow, begin by thinking about the type of data used (e.g., text or string data); the type of output desired (e.g., textual or image display); and the type of director needed to execute this model (e.g., synchronous or parallel) The "Hello World" workflow requires a constant actor, a text display actor, and a SDF director (in a SDF director, the data will flow through the actors based on the order in the workflow, and the workflow will run continuously). .

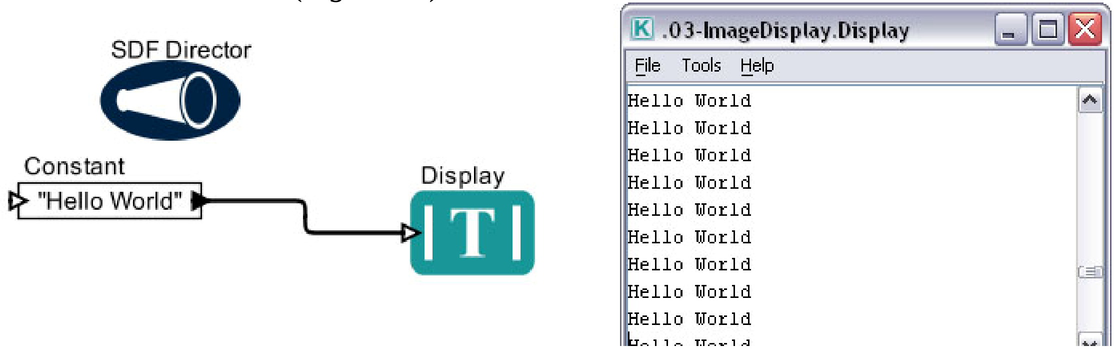
1. Open Kepler. A blank Workflow canvas will open.
2. In the Components and Data Access area, select the Components tab, then navigate to the "/Components/Director/" directory.
3. Drag the *SDF Director* to the top of the Workflow canvas.
4. In the Components tab, search for "Constant" and select the *Constant* actor.
5. Drag the *Constant* actor onto the Workflow canvas and place it a little below the *SDF Director*.
6. Configure the *Constant* actor by right-clicking the actor and selecting Configure Actor from the menu. (*Figure 15*)



```
value:          "Hello World"
class:          ptolemy.actor.lib.Const
semanticType00: urn:lsid:localhost:onto:1:1#ConstantActor
semanticType11: urn:lsid:localhost:onto:2:1#Constant
kar:            urn:lsid:kepler-project.org:kar:57:1
```

Figure 15: Configuring the *Constant* actor.

7. Type "Hello World" in the `value` field of the "Edit parameters for Constant" dialog window and click Commit to save your changes. "Hello World" is a string value. In Kepler, all string values must be surrounded by quotes. (*Figure 15*).
8. In the Components and Data Access area, search for "Display" and select the *Display* actor found under "Textual Output."
9. Drag the *Display* actor to the Workflow canvas.
10. Connect the output port of the *Constant* actor to the input port of the *Display* actor.
11. Run the model (*Figure 16*).



```
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
```

Figure 16: "Hello World" workflow and output.

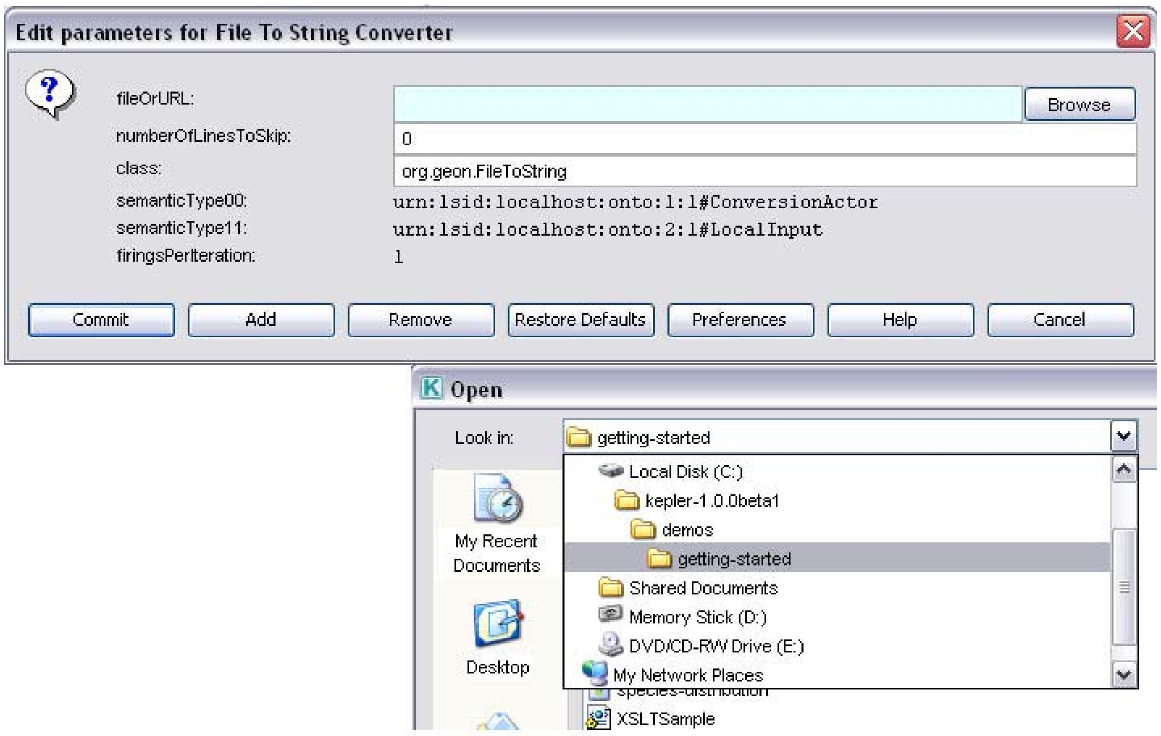
<u>NOTE</u>: By default, the SDF director will continuously run a workflow, creating a loop. To run "Hello World" a limited number of times, right-click on the SDF Director and select "Configure Director" from the menu. Type the desired number of iterations into the `iterations` field of the "Edit parameters for SDF Director" dialog window and click the Commit button to save your changes.

6.5.2. Example 2: Creating a Simple Workflow Using Local Data

In this example, you will create a simple workflow using an actor that will read a local data file containing information about species, extract the data from a column within that file, and then calculate count averages for each of the species.

Kepler can read data in many ways and from many formats: In this example, we will use an actor to review a data table. To determine which actor is appropriate, consider the format in which the data are saved. In this example, the data are saved in a text format. As such we will use the *File to String Converter* actor to view the data in a tabular format. This model requires two actors: a *File to String Converter* actor and a *Display* actor to output text. In addition, the example requires a *SDF Director*.

1. From the Menu bar, select File, then New Workflow, and then Blank. A new window will open with a blank Workflow canvas.
2. In the Components and Data Access area, select the Components tab, and then navigate to the "/Components/Director/" directory.
3. Drag the *SDF Director* to the top of the Workflow canvas.
4. In the Components tab, type "File to String Converter" in the Search box, then click the Search button.
5. Drag the *File to String Converter* actor onto the Workflow canvas and place it a little below the *SDF Director*.
6. Right-click the *File to String Converter* actor and select Configure Actor from the menu. An "Edit parameters for File to String Converter" dialog window will open.
7. Click the Browse button to the right of the `fileOrURL` parameter and navigate to the following file: mollusc_abundance.txt. These data come installed in Kepler and are located in the "/kepler/demos/getting-started/" folder.

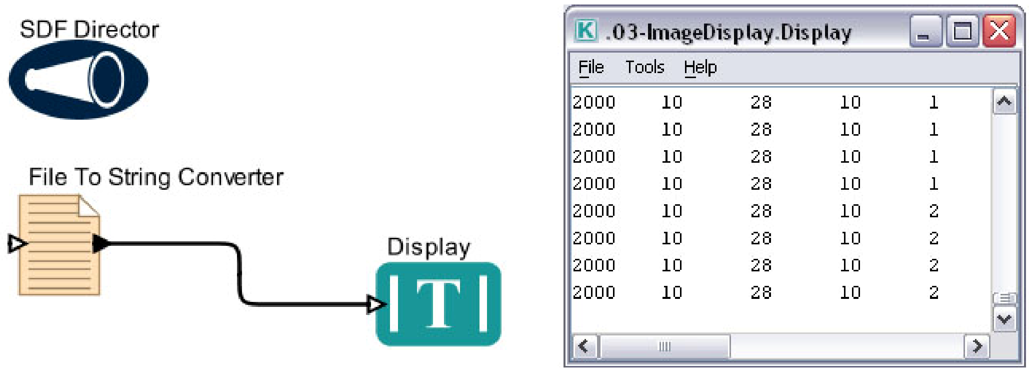

Edit parameters for File To String Converter

fileOrURL:	
numberOfLinesToSkip:	0
class:	org.geon.FileToString
semanticType00:	urn:lsid:localhost:onto:1:1#ConversionActor
semanticType11:	urn:lsid:localhost:onto:2:1#LocalInput
firingsPerIteration:	1

Figure 17: Configuring the *File to String Converter* actor to use data from your local machine.

8. Click the Commit button at the bottom of the "Edit Parameters for File to String Converter" dialog box. The actor is now configured to read the specified file.

9. In the Components tab, search for “Display”. Select the *Display* actor and drag it onto the Workflow canvas to the right of the *File to String Converter* actor.
10. Connect the output port of the *File to String Converter* actor to the input port of the *Display* actor.
11. From the Toolbar, select the Run button. A pop-up window will appear, displaying the contents of the data file in tabular format.
12. From the Menu bar, select File, then Save. When prompted, save the newly created workflow to the “/kepler/demos/getting-started” directory with the name “readingdata.xml.”



```
2000    10    28    10    1
2000    10    28    10    1
2000    10    28    10    1
2000    10    28    10    1
2000    10    28    10    2
2000    10    28    10    2
2000    10    28    10    2
2000    10    28    10    2
```

Figure 18: Using and displaying local data in a workflow.

<u>NOTE:</u> When creating a workflow, remember that the limitations of the data determine which processing components are appropriate.

7. Sample Scientific Workflows

This section examines a small set of sample scientific workflows that come standard with Kepler and provides step-by-step instructions for creating these workflows.

7.1. Sample Workflow 1 – Simple Addition

Name	Simple Addition Workflow
Filename	/kepler/demos/getting-started/01-SimpleAddition.xml
Detailed Description	This workflow adds two numbers together and displays the result. The *Constant* actors simply produce a constant value (3 and 2) that is sent to the *Add or Subtract* actor. The *Add or Subtract* actor calculates the sum of the input values, and then outputs that sum through its output port, which is connected to the *Monitor Value* actor. The *Monitor Value* actor displays the sum, in this case the value '5'. The workflow represents the process visually so that it is easy to understand how data flows from one component to another.
Assumptions	The *Constant* actor produces numeric values
Director	SDF Director
Data	Data is generated in the *Constant* actor

Actors	*Constant, Add or Subtract, Monitor Value*
Parameters	*Constant1*: `value`= 2 *Constant2*: `value`=3

The Simple Addition workflow adds two numbers together and displays the results. To create this workflow:

1. In the Components and Data Access area, select the Components tab.
2. Search for the *SDF Director* and drag and drop the director to the Workflow canvas. Search for the *Constant* actor and drag and drop that to the screen twice. Note the second actor is named *Constant2*.
3. Configure the *Constant1* actor by right-clicking the actor and selecting Configure Actor. In the "Edit Parameters for Constant" window, add 2 to the `value` field and click Commit.
4. Configure the *Constant2* actor to use 3 as the `value` parameter.
5. Search for the *Add or Subtract* actor, and drag and drop that to the screen.
6. Connect the two *Constant* actors to the *Add or Subtract* actor. To add the two values, both the *Constant1* and *Constant2* actor must be connected to the + (top) input port of the *Add or Subtract* actor.
7. Search for the *Monitor Value* actor, and drag and drop that to the screen.
8. Connect the output of the *Add or Subtract* actor to the input of the *Monitor Value* actor.

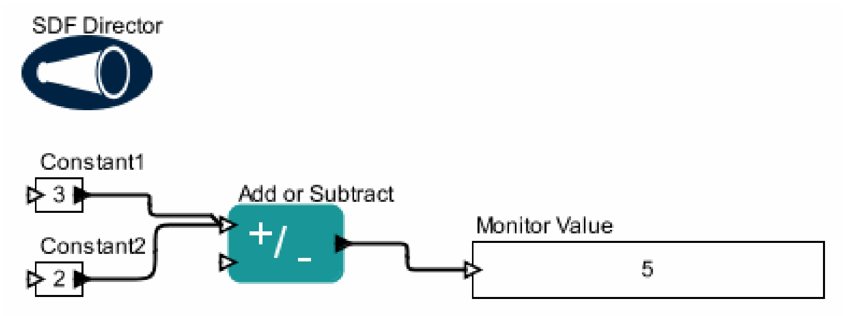


Figure 19: The Simple Addition workflow and its output

7.2. Sample Workflow 2 –Linear Regression

Name	Simple Linear Regression workflow using R
Filename	/kepler/demos/getting-started/05-LinearRegression.xml
Detailed Description	This workflow performs a simple linear regression analysis using the *RExpression* actor. The workflow creates a scatter plot of the two variables from the Datos Meteorologicos data set and adds a regression line using the ***Y = a + bX equation***, where ***X*** is the explanatory variable and ***Y*** is the dependent variable. The slope of the line is ***b***, and ***a*** is the intercept (the value of ***y*** when ***x*** = 0).
Assumptions	A linear regression assumes linearity, independence, homoscedasticity, and normality

Director	SDF director
Data	Datos Meteorologicos
Actors	*Datos Meteorologicos, RExpression, Display, ImageJ*

Parameters	*Datos Meteorologicos*: `Data Output Format` = As Column Vector
	
	SDF Director: `iterations` = 1;
	
	RExpression: `R function or script` =
	
	`res <- lm(BARO ~ T_AIR)`
	
	`res`
	
	`plot(T_AIR, BARO)`
	
	`abline(res);`
	
	RExpression: input ports = ‘T_AIR’ and ‘BARO.’

The Simple Linear Regression workflow runs a search for data on the EcoGrid, and the data found is used to create a workflow conducting a linear regression. In this example, the input data comes from two output ports (the data columns on Barometric Pressure and Air Temperature) of the *Datos Meteorologicos* actor, a data set of meteorological data from the La Hechicera station collected in 2001.

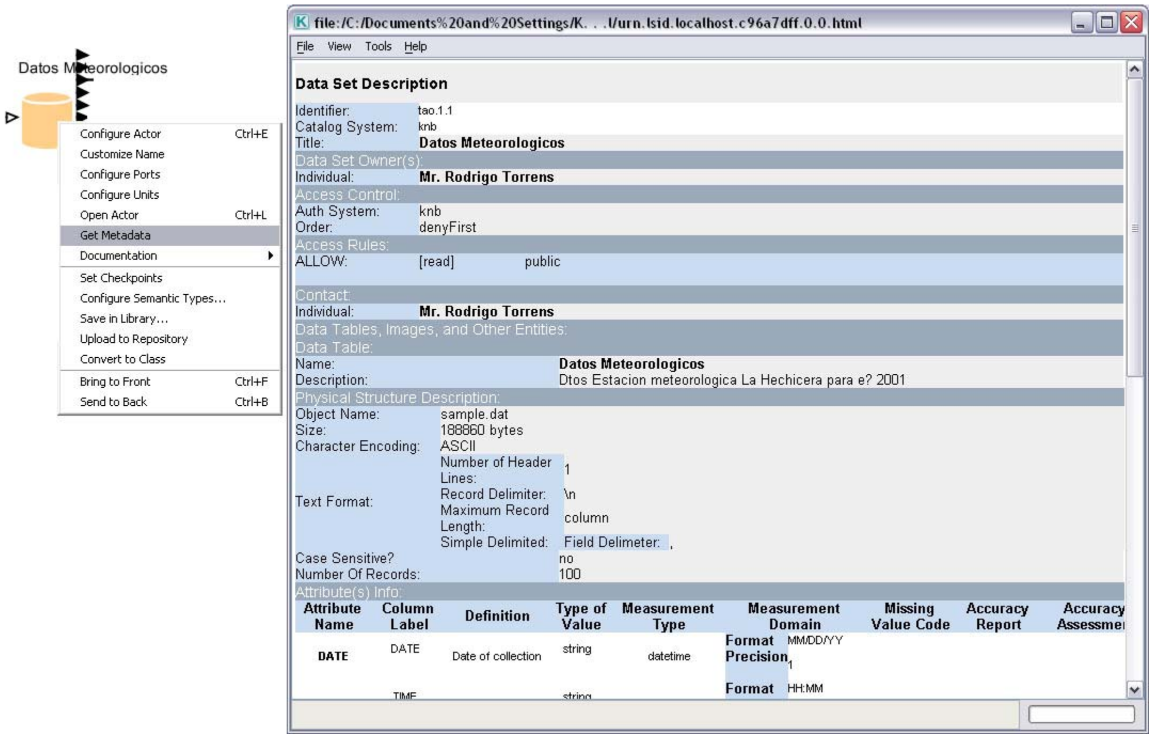
The Linear Regression workflow uses four actors (the *Datos Meteorologicos* actor, the *RExpression* actor, the *ImageJ* actor and the *Display* actor) and the *SDF Director*. The *RExpression* actor inserts R commands and scripts into the workflow. The *RExpression* actor makes integrating the powerful data manipulation and statistical functions of R into workflows easy. To implement the *RExpression* actor, R must be installed on the computer running the Kepler application.

<u>NOTE:</u> If you have problems creating this workflow, a stored version comes standard with Kepler at kepler/demos/getting-started/05LinearRegression.xml.

To create the Simple Linear Regression workflow:

1. Select the Data tab in the Components and Data pane.
2. Click the Sources button and limit the scope of the search by unchecking “KU Digir EcoGrid QueryInterface” and “GEON Search QueryInterface.” Because *Datos Meteorologicos* is stored on the KNB Metacat, the data source for the search can be limited to just those nodes on the grid.
3. Click Ok to confirm and store the search source changes.
4. Type *Datos Meteorologicos* in the search box and click Search.
5. From the search results, click the *Datos Meteorologicos* icon. Drag and drop the *Datos Meteorologicos* actor to the Workflow canvas.

<u>NOTE</u>: To find more information about the data set, right-click *Datos Meteorologicos* in the Components and Data Access area and select Get Metadata. Depending upon the amount of information entered by the provider, much valuable metadata can be obtained. For the Datos Meteorologicos data set, use the Attribute Name (e.g., BARO and T_AIR) to read and incorporate data into the R script. The type of value and measurement type of each attribute help you decide which statistical models are appropriate to run.



Data Set Description

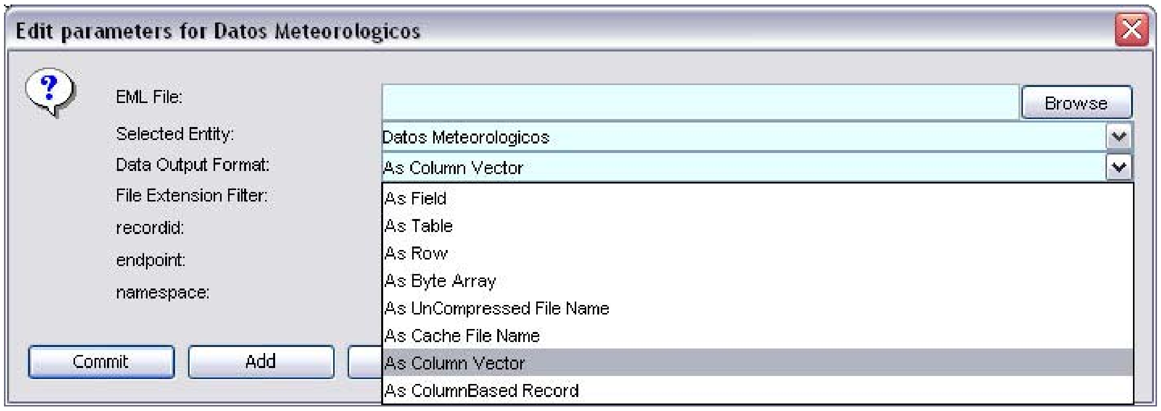
Identifier:	tao.1.1
Catalog System:	knb
Title:	**Datos Meteorologicos**
Data Set Owner(s):	
Individual:	**Mr. Rodrigo Torrens**
Access Control:	
Auth System:	knb
Order:	denyFirst
Access Rules:	
ALLOW:	[read] public
Contact:	
Individual:	**Mr. Rodrigo Torrens**
Data Tables, Images, and Other Entities:	
Data Table:	
Name:	**Datos Meteorologicos**
Description:	Dtos Estacion meteorologica La Hechicera para e? 2001
Physical Structure Description:	
Object Name:	sample.dat
Size:	188860 bytes
Character Encoding:	ASCII
Text Format:	Number of Header Lines: 1; Record Delimiter: \n; Maximum Record Length: column; Simple Delimited: Field Delimeter: ,
Case Sensitive?	no
Number Of Records:	100

Attribute(s) Info:

Attribute Name	Column Label	Definition	Type of Value	Measurement Type	Measurement Domain	Missing Value Code	Accuracy Report	Accuracy Assessme
DATE	DATE	Date of collection	string	datetime	**Format** MM/DD/YY **Precision** 1			
	TIME		string		**Format** HH:MM			

Figure 20: ViewingMetadata

5. Right-click the *Datos Meteorologicos* actor and select Configure Actor. Select "As Column Vector" from the pull-down menu beside the `Data Output Format` parameter and click Commit. (The data type of the *Datos Meteorologicos* actor must be set to "As Column Vector" to match the *RExpression* actor)


Edit parameters for Datos Meteorologicos

EML File:

Selected Entity: Datos Meteorologicos

Data Output Format: As Column Vector

File Extension Filter:

recordid:

endpoint:

namespace:

Figure 21: Configuring *Datos Meteorologicos*

<u>NOTE:</u> *Datos Meteorologicos* has a series of output ports corresponding to the data attribute names (e.g., BARO and T_AIR). To locate the appropriate port, mouse-over the output ports and review the port tooltips.

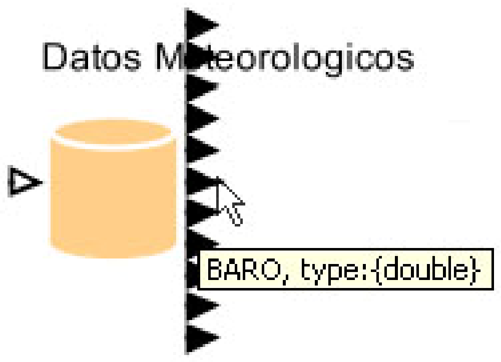


Figure 22: Locating data ports

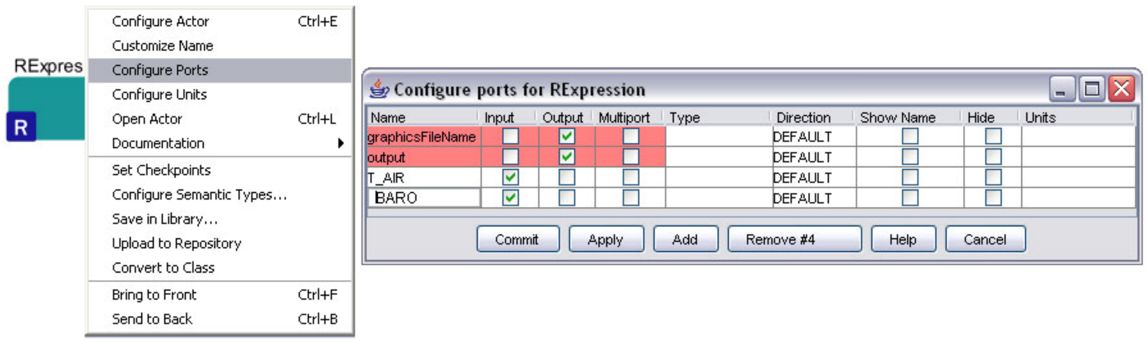
To finish creating the workflow, add the *SDF Director* and the remaining actors (*RExpression, ImageJ, Display).*

7. Locate the *SDF Director* and drag and drop that to the Workflow canvas.
8. Configure the *SDF Director* by right-clicking the actor and selecting Configure Actor. Change the number of `iterations` to 1.
9. Click Commit for the changes to take effect.
10. Locate the *RExpression* actor and drag and drop it to the Workflow canvas. The *RExpression* actor is located in the “Mathematical Operator” folder.

By default, the *RExpression* actor is configured with two output ports and the R script 2+2. Before you can use the *RExpression* actor in the Simple Linear Regression workflow, you must add two input ports (T_AIR and BARO) and reconfigure the *RExpression* script.

11. Right-click the *RExpression* actor and select Configure Ports.

12. In the “Configure ports” dialogue box, click Add twice to add two new ports. Designate the new ports as input ports by clicking the checkbox named Input beside each port.
13. Name the new input ports by double-clicking the blank box in the Name column. Add the name “T_AIR” for one input and “BARO” for the other. Click Commit to save the changes.



Name	Input	Output	Multiport	Type	Direction	Show Name	Hide	Units
graphicsFileName	☐	✓	☐		DEFAULT	☐	☐	
output	☐	✓	☐		DEFAULT	☐	☐	
T_AIR	✓	☐	☐		DEFAULT	☐	☐	
BARO	✓	☐	☐		DEFAULT	☐	☐	

Figure 23: Adding and customizing ports

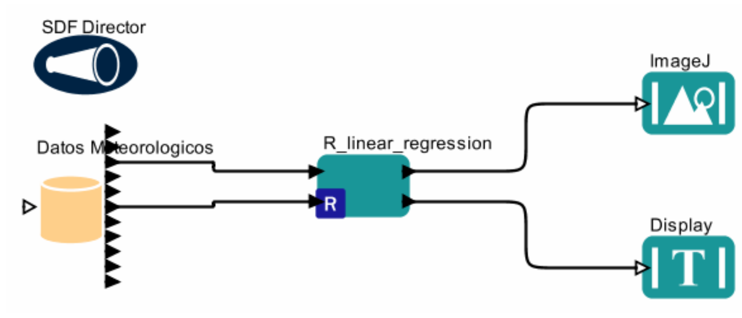
14. To configure the R script, right-click the *RExpression* actor and select Configure Actor. In the “Edit parameters” dialogue box, change the value of the `R function or script` from `(2+2)` to the following:

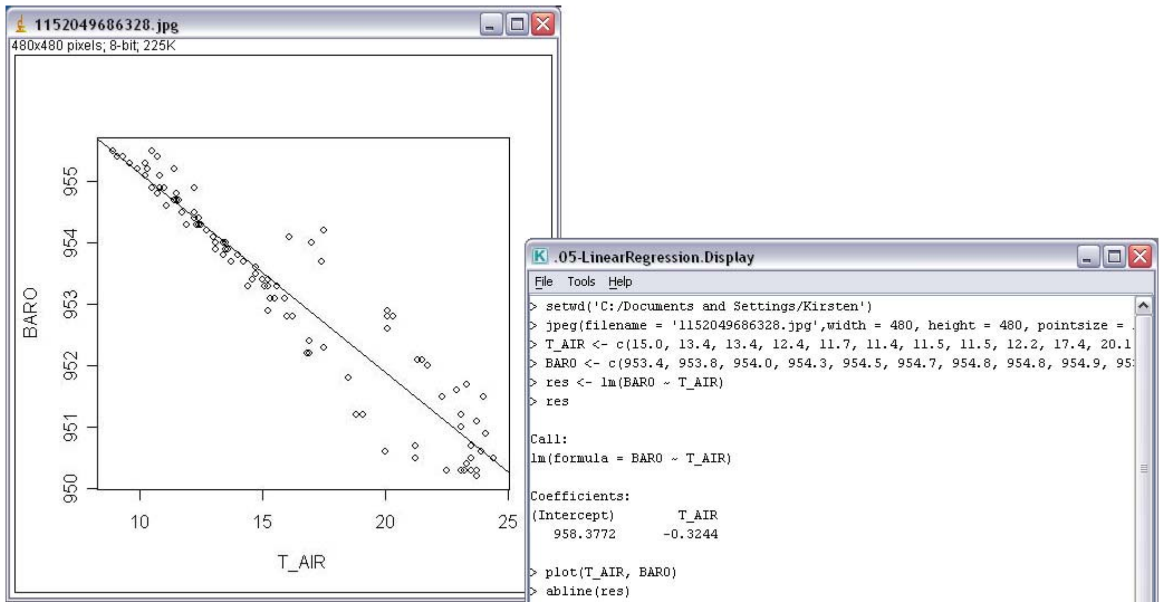
```
        res <- lm(BARO ~ T_AIR)

        res

        plot(T_AIR, BARO)

        abline(res)
```

The above R function tells the *RExpression* actor to read the Barometric Pressure and Air Temperature data and then plot the values along with a regression line. Click Commit to save your changes.

15. Drag and drop the text *Display* actor to the Workflow canvas. The *Display* actor is located under “Components> Data Output > Workflow Output > Textual Output.”
16. Connect the lower output port of the *RExpression* actor to the input port on the *Display* actor.
17. Drag and drop the *ImageJ* actor to the Workflow canvas, The *ImageJ* actor is located under “Components > Data Output > Workflow Output > Graphical Output.”

Connect the upper output port of the *RExpression* actor to the input port of the *ImageJ* actor. You are now ready to run the workflow. The resulting workflow and graphic output are shown below.





```
> setwd('C:/Documents and Settings/Kirsten')
> jpeg(filename = '1152049686328.jpg',width = 480, height = 480, pointsize =
> T_AIR <- c(15.0, 13.4, 13.4, 12.4, 11.7, 11.4, 11.5, 11.5, 12.2, 17.4, 20.1
> BARO <- c(953.4, 953.8, 954.0, 954.3, 954.5, 954.7, 954.8, 954.8, 954.9, 95
> res <- lm(BARO ~ T_AIR)
> res

Call:
lm(formula = BARO ~ T_AIR)

Coefficients:
(Intercept)        T_AIR
   958.3772      -0.3244

> plot(T_AIR, BARO)
> abline(res)
```

Figure 24: Linear Regression workflow and its output

The left-hand window in *Figure 24* displays the scatter plot of Barometric pressure to Air Temperature along with a regression line. The graph shows a strong negative relationship between the two: as air temperature lowers, the Barometric pressure rises. The right-hand window displays the Barometric Pressure and Air Temperature data used in the scatter plot. Additionally, the intercept on the Y-axis (958.38 Barometric Pressure and the slope – 0.32 for the linear regression equation y=mx+b) is displayed.

You may change the data type and the data set that is run through the workflow. When changing the data, remember to make sure that the data meets the assumptions mentioned in workflow table at the beginning of Section 7.2.

7.3. Sample Workflow 3 – Web Services and Data Transformation

Name	Web Services and Data Transformation Workflow
Filename	06-WebServicesAndDataTransformation.xml
Detailed Description	This workflow uses the remote genomics data service to retrieve a genetic sequence for a given gene accession

	number. The sequence is then displayed in three different ways after appropriate transformations: first in its native format (XML), second as a sequence of elements extracted from the XML format, and third as an HTML document that can be used for display on a website. The later two operations are performed using Composite actors that hide some of the complexity of the underlying operations. Composite actors can be thought of as "sub-workflows" that execute a potentially complex set of tasks with a single actor.
Assumptions	The *Web Service* actor assumes that the target Web service is RPC-based and uses primitive XML types and arrays.
Director	*SDF Director*
Data	The data consists of an initial input gene accession number that is specified by the *String Constant* actor and an intermediate input retrieved from the remote genomics data service.
Actors	*String Constant, Display, Sequence Getter Using XPath, HTML Generator Using XSLT, Web Services,*
Parameters	*Web Services*: `wsdlUrl`=http://xml.nig.ac.jp/wsdl/DDBJ.wsdl *Web Services*: `methodName`=getXMLEntry

The Web Services and Data Transformation workflow uses the *Web Service* actor to access a genomics database and return a genetic sequence from it, which is queried using a remote genomics data service. The name of the returned genetic sequence (i.e., the gene accession number) is passed to the *Web Services* actor by a *String Constant* actor. The *Web Service* actor must be configured to access the appropriate remote server. Once configured, the *Web Service* actor outputs the gene sequence obtained from the remote server so that it can be displayed in multiple formats using three different textual *Display* actors: one for XML (the format in which the results are returned by default), one for a sequence of elements extracted from the XML, and one for an HTML document that can be displayed on a website.

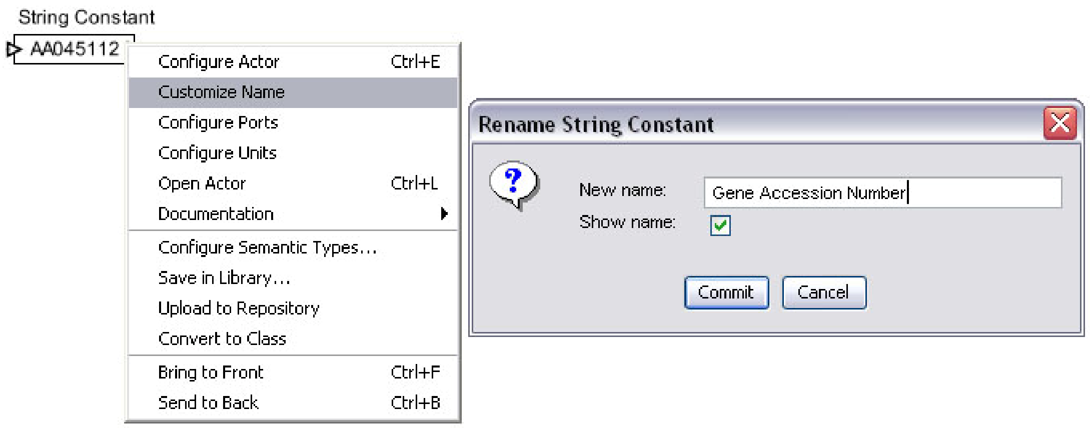
A Relation is used to "split" the data output by the *Web Service* actor so that it can be shared by all of the necessary components.

The workflow uses two composite actors: *Sequence Getter Using XPath* and *HTML Generator Using XSLT* to process the returned XML data and convert it into a sequence of elements and an HTML file, respectively. These actors have been created for use with this workflow using the existing Kepler actors. *Sequence Getter Using XPath* and *HTML Generator Using XSLT* do not appear in the Components tab. To see the "insides" of the composite actors, right-click the actor icon on the Workflow canvas and select Open Actor from the menu. The composite actor will open in a new application window. See *Figure 28* for an example.

In addition, the workflow uses a fourth *Display* actor to display errors returned by the remote server (e.g., server down or incorrect input).

To create the Web Services workflow:

1. Open a new Workflow canvas.
2. Drag and drop the *SDF Director* onto the Workflow canvas.
3. Drag and drop the *String Constant* actor onto the Workflow canvas.
4. Right-click the *StringConstant* actor and select Configure Actor. Type AA045112 (the gene accession number) into the `value` field and click Commit.
5. To change the name of the *String Constant* actor, right-click it and select Customize Name. Type a new name (e.g., Gene Accession Number) into the New name field and click Commit.

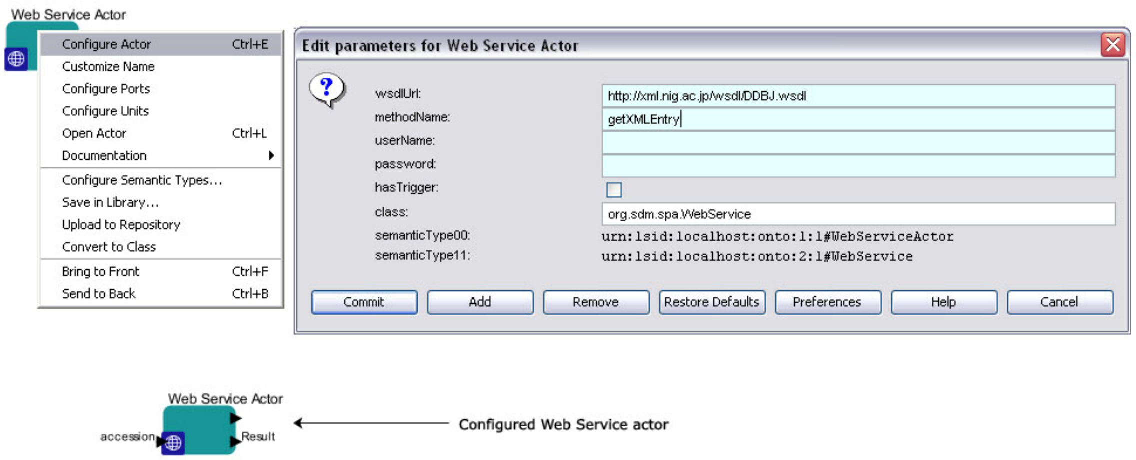


New name: Gene Accession Number
Show name: ✓

Figure 25: Customizing the name of an actor

6. Drag and drop the *Web Service* actor onto the Workflow canvas. Place the actor beneath the *String Constant* actor. By default, the *Web Service* actor has one output port for displaying runtime errors and must be configured with a Web service URL (a `wsdlUrl` parameter), an appropriate method (a `methodName` parameter). Once the actor has been configured with this information, it will automatically generate the correct input and output ports required by the Web service.
7. To configure the parameters required for accessing the Web service, right-click the *Web Service* actor and select Configure Actor. Type http://xml.nig.ac.jp/wsdl/DDBJ.wsdl into the `wsdlUrl` field. In the

`methodName` field, type `getXMLEntry`. Click commit. The Web Service actor ports should update automatically. You can move the ports so that they are more conveniently located by right-clicking the actor and selecting a desired port direction from the Configure Ports dialog box.



wsdlUrl: http://xml.nig.ac.jp/wsdl/DDBJ.wsdl
methodName: getXMLEntry
userName:
password:
hasTrigger: ☐
class: org.sdm.spa.WebService
semanticType00: urn:lsid:localhost:onto:1:1#WebServiceActor
semanticType11: urn:lsid:localhost:onto:2:1#WebService

Figure 26: Configuring the *Web Service* actor.

8. Connect the output of the *StringConstant* actor (Gene Accession Number) to the input of the *Web Service* actor.
9. Drag and drop four *Display* actors onto the Workflow canvas.
10. Position one of the *Display* actors beneath and to the right of the *Web Service* actor. Right-click the actor and change the name to “Errors Sink.”
11. Connect the lower output port of the *Web Service* actor to the input port of the “Errors Sink” *Display* actor.
12. Position the second *Display* actor to the right and slightly above the *Web Service* actor. Right-click the actor and change the name to “XML Entry Display.”

The Web Services and Data Transformation workflow uses two component actors designed specifically for this workflow. These customized actors are not available in the Component library, and rather than recreating them, we will save some time by copying and pasting them from the existing workflow.

13. Open the Web Services and Data Transformation workflow (06-WebServicesAndDataTransformation.xml). The workflow will open in a new application window. Select the *Sequence Getter Using XPath* composite actor by left-clicking it.
14. From the Edit menu, select Copy (or use the keyboard shortcut Ctrl+C).
15. Return to your in-progress workflow and paste the *Sequence Getter Using XPath* actor to the right of the *Web Service* actor using the Paste command available in the Edit menu or the keyboard shortcut Ctrl+V.

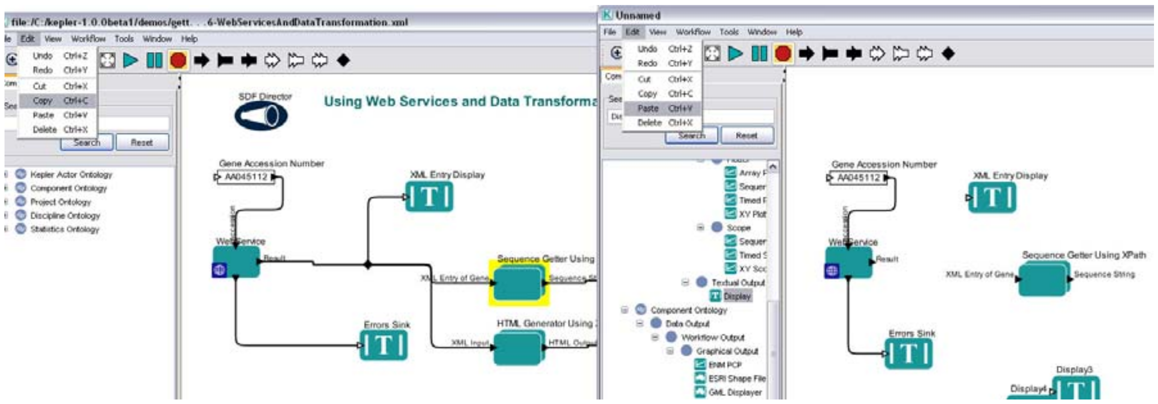


Figure 27: Copying and pasting actors between workflows.

16. Copy and paste the *HTML Generator Using XSLT* actor from the Web Services and Data Transformation workflow into your in-progress workflow.

<u>NOTE:</u> To view the inner workings (i.e., hierarchical layers) of a composite actor, right-click the actor and select Open Actor from the menu. The composite actor will open in a new application window. Composite actors can be thought of as "sub-workflows" that execute a potentially complex set of tasks with a single actor.

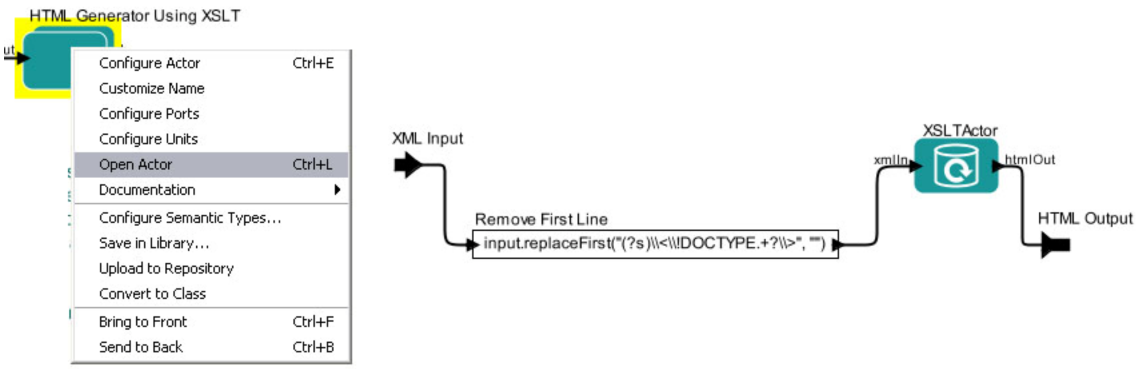


Figure 28: Inside the *HTML Generator Using XSLT* composite actor.

Because the *Web Services* output is required by three actors, before connecting your actors, you must add a relation to direct the output to multiple ports.

17. Add a relation by clicking the Relation icon at the far right of the Toolbar. The relation (represented by a dark diamond icon) will appear near the center of the Workflow canvas. (You can also add a relation with the keyboard shortcut Ctrl-click).

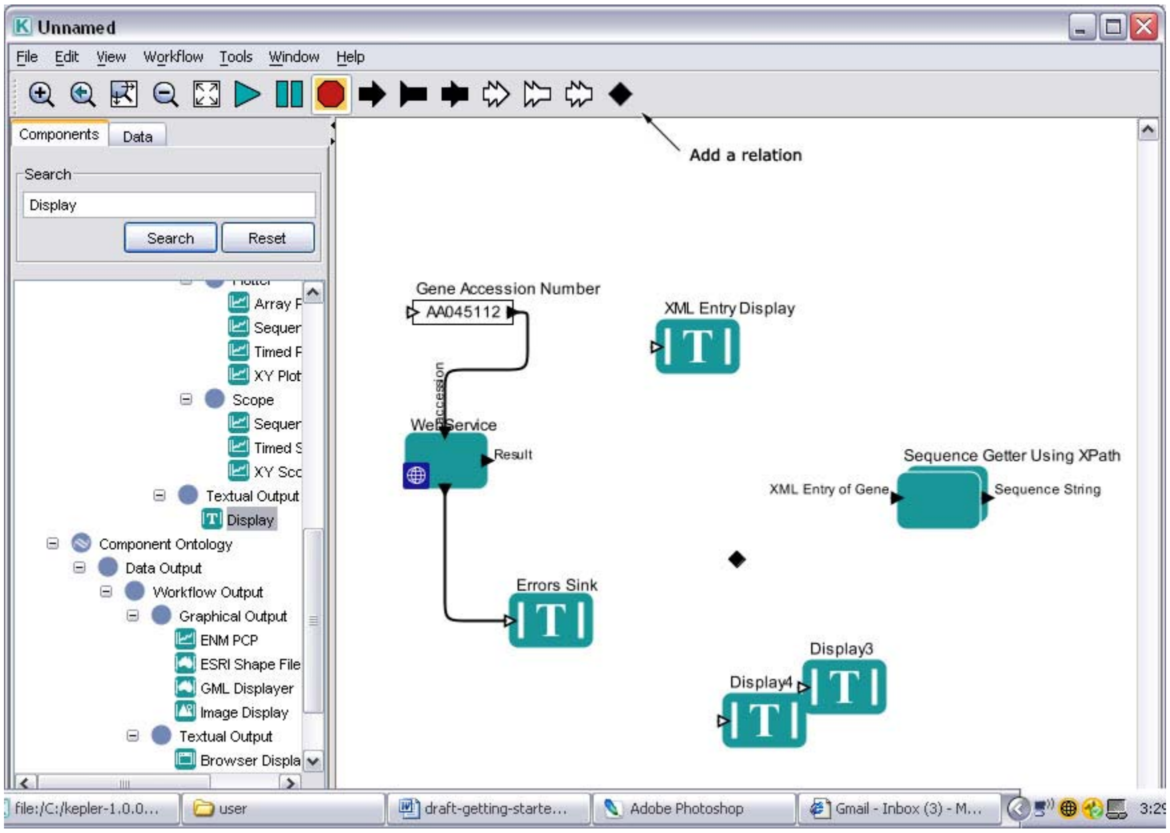
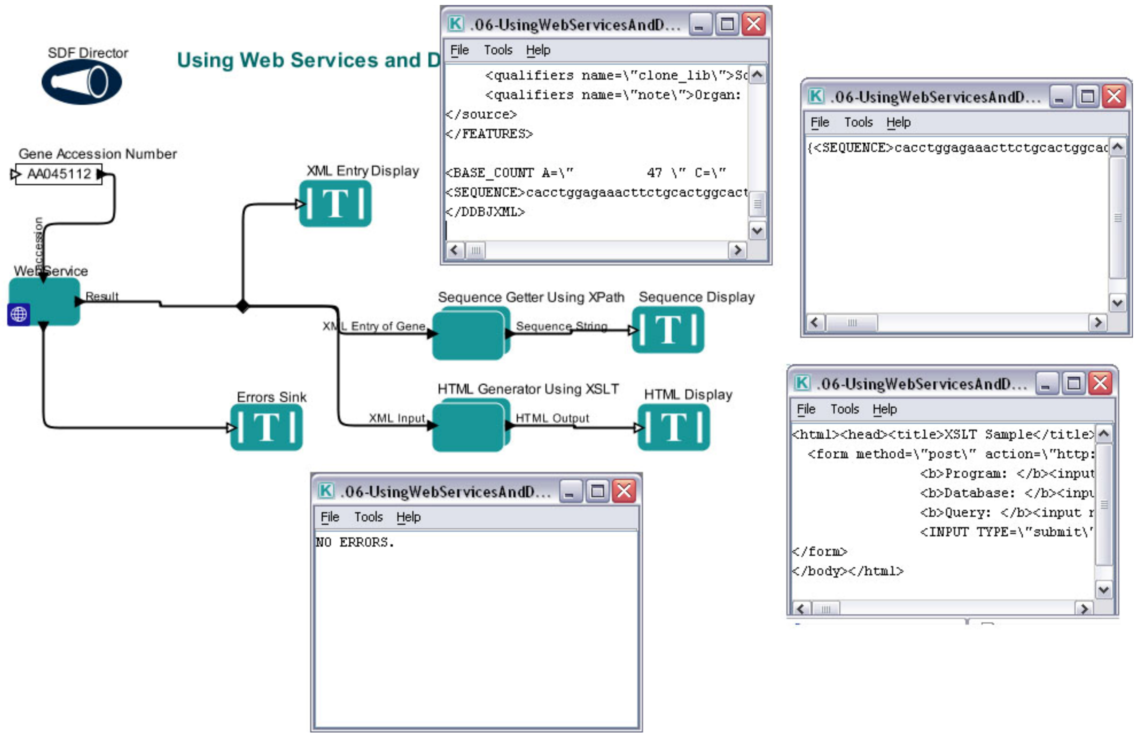


Figure 29: Adding a relation

18. Position the Relation icon between the *Web Service* actor and the *Sequence Getter using XPath* actor.
19. Connect the input port of the “XML Entry Display” *Display* actor to the output of the *Web Service* actor. To make the connection, start from the input port of the *Display* actor and drag the cursor to the center of the Relation icon.
20. Connect the *HTML Generator Using XSLT* actor and the *Sequence Getter Using XPath* actor to the Relation icon as well.
21. Rename the third *Display* actor “Sequence String Display” and position it to the right of the *Sequence Getter using XPath* actor.
22. Connect the input of the “Sequence String Display” actor to the output of the *Sequence Getter using XPath* actor.
23. Rename the fourth *Display* actor “HTML Display” and position it to the right of the *HTML Generator Using XSLT* actor.
24. Connect the input of the “HTML Display” actor to the output of the *HTML Generator Using XSLT* actor.

You are now ready to run the workflow. The resulting workflow and output are shown below.



```
    <qualifiers name=\"clone_lib\">Sc
    <qualifiers name=\"note\">Organ:
</source>
</FEATURES>

<BASE_COUNT A=\"          47 \" C=\"
<SEQUENCE>cacctggagaaacttctgcactggcact
</DDBJXML>
```

```
{<SEQUENCE>cacctggagaaacttctgcactggcac
```

```
<html><head><title>XSLT Sample</title>
  <form method=\"post\" action=\"http:
                <b>Program: </b><input
                <b>Database: </b><inpu
                <b>Query: </b><input r
                <INPUT TYPE=\"submit\'
</form>
</body></html>
```

```
NO ERRORS.
```

Figure 30: The Web Services workflow

NOTE: To add an annotation to your workflow, drag-and-drop the *Annotation* actor onto the Workflow canvas. Double-click the default text ("Double click to edit") to customize the annotation.

8. Appendix

8.1. Ptolemy II – The Foundation of Kepler

Ptolemy II is a software framework for heterogeneous, concurrent modeling and design, with a Java-based component assembly framework using a graphical interface called Vergil. The Ptolemy II software is a product of the Ptolemy project at the University of California at Berkeley, a project whose goal is "the use of well-defined models of computation that govern the interactions between components."

As explained at the project's website, Ptolemy II includes a number of *domains*, each of which realizes a model of computation. It also includes a component library and a number of support packages such as graphing, mathematics, plot, and data packages. For more information about Ptolemy II, see http://ptolemy.eecs.berkeley.edu/index.html.

Although not originally intended for scientific workflows, Ptolemy II provides support for dataflow-oriented models, which is a very important characteristic of scientific workflows. Because Ptolemy II provides an open-source, mature platform for model design and execution, including various models of computation, and is well documented and easily extensible, it was chosen as the foundation for Kepler.

8.2. Technical Overview of Kepler

Kepler is an open-source software tool for scientists to design and execute scientific workflows efficiently using emerging Grid-based approaches to distributed computation. Kepler is based on the Ptolemy II software and inherits much of its functionality, with particular emphasis on:

- model design and execution;
- synchronous dataflow computational model;
- process networks computational model;
- extensibility; and
- polymorphism.

From a technical perspective, the Kepler software adds:

- Web services;
- grid-based services;
- MoML (modeling markup language – an XML description of a Kepler workflow);
- ontology based processing (under development); and
- semantic-type checking (under development).

Kepler provides access to local and remote data through Web and grid services. It provides a component that instantiates itself to perform remote calls as if they were running on the local machine. Kepler also offers a large library of Java-based components to display, manipulate, and process data as it moves through execution in a scientific workflow. Programmers can "wrap" external functionality in a Java wrapper so that it can be added to the component library. Programmers can also create new components using Java that can be used in Kepler.

Kepler uses a director/actor metaphor. Models of computation (directors) coordinate and direct the execution of the components (actors) of a scientific workflow. Actors can represent data or processing components.

Directors

A director governs the execution within a workflow or Composite actor. A composite actor that contains a director is said to be *opaque*, and the execution model within the composite actor is determined by the contained director. This director is called the *local director* of a composite actor. A composite actor is also aware of the director of its container, which is referred to as its *executive director*. If a composite actor lacks an explicit local director, then the execution model for that actor is determined by its executive director (i.e., the director of the composite in which it is contained).

Actors

Actors are re-useable components that execute a variety of functions and communicate with other actors in a workflow. Actors communicate through ports. Messages (input/output data) are encapsulated in tokens. Messages are sent through ports.
An atomic actor is an executable entity that cannot itself contain other actors. The ports of atomic actors are constrained to be IOPorts. Derived classes may further constrain the ports by overriding the public method newPort() to create a port of the appropriate subclass, and the protected method _addPort() to throw an exception if its argument is a port that is not of the appropriate subclass. In this base class, the actor does nothing in the action methods.

A Composite actor is an aggregation of actors. It may have a *local director*, which is responsible for executing the contained actors. At the top level of a hierarchy, a composite actor (the top level Composite actor of the topology) will normally exist with a local director, and no container. A Composite actor at a lower level of the hierarchy may also have a local director. A Composite actor with a local director is *opaque*, and serves the role of the *wormhole* from Ptolemy Classic. Its ports are opaque, but it can contain actors and relations. The top-level composite actor is also associated with a manager object that is responsible for managing any execution within the topology at a high level

Parameters
Parameters are named values that can be attached to a workflow or to individual directors or actors. When you select the "Configure Actor" menu (or double-click on the actor) an "Edit Parameters" dialog appears containing parameters for the actor that can be specified by entering a value. Parameters can also be added directly to a workflow. The name and value can be specified, and the name can then be used elsewhere in the workflow. For example, one could create a parameter named 'Count" with a value of '10'. Then in any actors on the workspace that need a count, one can enter the name 'Count' rather than the explicit value of '10'. This is particularly useful when one want to change a parameter value, as the change needs to be made in only one place.

Created parameters can be used as a secondary method for providing data to workflow actors. They may be thought of a 'global variables'; i.e., data that does not travel through the ports. Parameters can also be used in algebraic expressions as a way of providing even more complex data.

Ports
A port is the interface of an entity to any number of relations. Normally, a port is contained by an entity, although a port may exist with no container. The role of a port is to aggregate a set of links to relations. Thus, for example, to represent a directed graph, entities can be created with two ports, one for incoming arcs and one for outgoing arcs. More generally, the arcs to an entity may be divided into any number of subsets, with one port representing each subset.

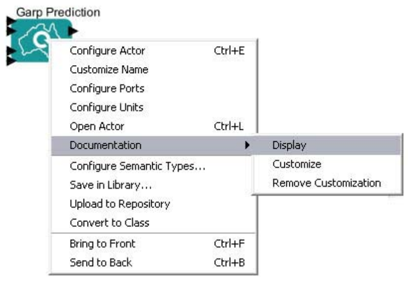
A port can link to any instance of relation. Derived classes may wish to constrain links to a subclass of relation. To do this, subclasses should override the protected method checkLink(Relation) to throw an exception if its argument is a relation that is not of the appropriate subclass. Similarly, if a subclass wishes to constrain the containers of the port to be of a subclass of entity, they should override the protected method _checkContainer(Entity).

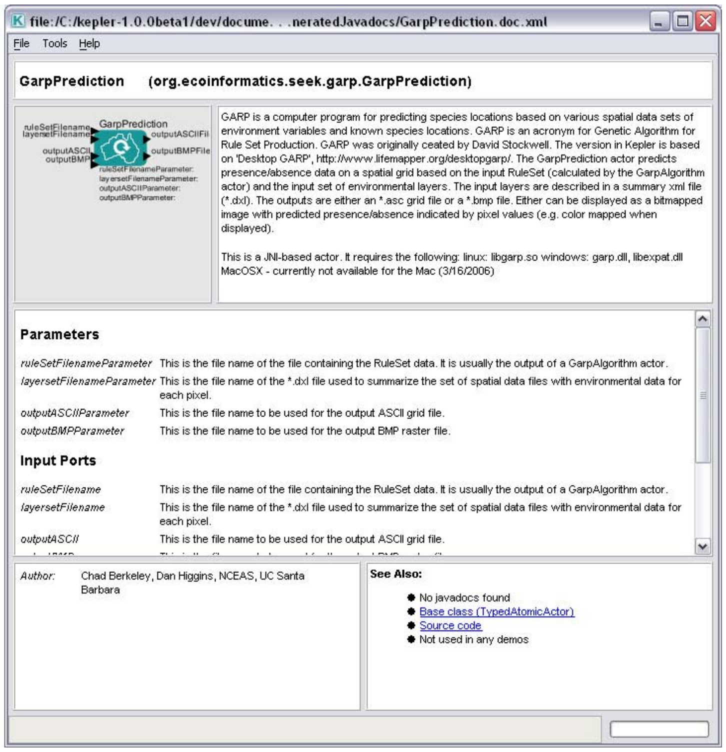
Relations
A relation is an object representing an interconnection between entities used to broadcast the output from a single port to a number of places. The single port still has only one connection to it, a connection to a relation. Relations can also be used to control the routing of wires in the diagram. A relation is represented in the diagram by a black diamond and can be selected by clicking the black diamond button in the toolbar.

8.3. Actor Reference

Documentation for actors and directors is located at http://www.kepler-project.org/nightly/docResults/generatedJavadocs. Additionally, this documentation is available within the Kepler interface. To get documentation:

1. Right-click the actor or director
2. Select Documentation
3. Then select Display.





GarpPrediction (org.ecoinformatics.seek.garp.GarpPrediction)

GARP is a computer program for predicting species locations based on various spatial data sets of environment variables and known species locations. GARP is an acronym for Genetic Algorithm for Rule Set Production. GARP was originally ceated by David Stockwell. The version in Kepler is based on 'Desktop GARP', http://www.lifemapper.org/desktopgarp/. The GarpPrediction actor predicts presence/absence data on a spatial grid based on the input RuleSet (calculated by the GarpAlgorithm actor) and the input set of environmental layers. The input layers are described in a summary xml file (*.dxl). The outputs are either an *.asc grid file or a *.bmp file. Either can be displayed as a bitmapped image with predicted presence/absence indicated by pixel values (e.g. color mapped when displayed).

This is a JNI-based actor. It requires the following: linux: libgarp.so windows: garp.dll, libexpat.dll MacOSX - currently not available for the Mac (3/16/2006)

Parameters

ruleSetFilenameParameter	This is the file name of the file containing the RuleSet data. It is usually the output of a GarpAlgorithm actor.
layersetFilenameParameter	This is the file name of the *.dxl file used to summarize the set of spatial data files with environmental data for each pixel.
outputASCIIParameter	This is the file name to be used for the output ASCII grid file.
outputBMPParameter	This is the file name to be used for the output BMP raster file.

Input Ports

ruleSetFilename	This is the file name of the file containing the RuleSet data. It is usually the output of a GarpAlgorithm actor.
layersetFilename	This is the file name of the *.dxl file used to summarize the set of spatial data files with environmental data for each pixel.
outputASCII	This is the file name to be used for the output ASCII grid file.

Author: Chad Berkeley, Dan Higgins, NCEAS, UC Santa Barbara

See Also:
- No javadocs found
- Base class (TypedAtomicActor)
- Source code
- Not used in any demos

Figure 31: Actor documentation