Appendix B

Writing Plug-Ins

B.1 Introduction

If you are experienced with Java, you can write a plug-in for HEC-DSSVue to provide custom capabilities for your office. Interfacing a plug-in is rather simple once you have written your own custom code. To do so, you copy existing plug-in source code and modify it so that it calls your custom code, call a registration function, compile and add the class files to a plug-in jar, and then change the manifest to show your class. To access your plug-in, simply copy the jar file to the Plugins directory and run HEC-DSSVue.

B.2 Code Overview

The "main" function of a plug-in is relatively simple; it just creates a menu or toolbar item, adds an action associated with that item, and registers it with the main class, *ListSelection*. We'll first work with the skeleton plug-in GenericDssVuePlugin.

```
public class GenericDssVuePlugin {

    public static void main(Object[] args)
    {
        final GenericDssVuePlugin plugin = new GenericDssVuePlugin();
        final ListSelection listSelection = (ListSelection) args[0];

        JMenuItem genericMenuItem = new JMenuItem("Generic Plugin");
        genericMenuItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e){
                plugin.process(listSelection);
            }
        });
       listSelection.registerPlugin(ListSelection.TOOLS, genericMenuItem);
     }
```

In process, you tell ListSelection to read the selected data:

```
protected void process(ListSelection listSelection) {
   List[] dataList = listSelection.getSelectedDataContainers();
   List tsContainers = dataList[0];
   boolean success = processData(listSelection, tsContainers);
}
```

B.3 Procedure

To write a plug-in you need Java and a sample plug-in to start with. The GenericDssVuePlugin is included with the program in the Samples directory. The source code is located in that jar, along with the class files and manifest.

You will need to refer to the functions listed in the Scripting Chapter. These include documentation of what is in a TimeSeriesContainer class, how to use HecTime, and various ListSelection calls. All of the scriptable calls are available for use in plug-ins.

1. You need a Java IDE (Integrated Development Environment), but the examples shown will just be using the J2SE (Java 2 Standard Edition) SDK (Software Development Kit). You can download the SDK from sun at: http://java.sun.com/j2se/. You can get an excellent Java IDE for free named Eclipse from: http://www.eclipse.org/.
2. From the GenericDssVuePlugin.jar file, extract "GenericDssVuePlugin.java" to a java file named according to your task. You can put it in any directory you desire if you are not going to use a package name. If you use a package name, then you have to put it into a directory that is the same as the package name.
3. Edit the .java file you copied:
   a). Change the class name and the name in the main to the task name you gave the .java file. The examples do not specify a package or directory, but you may want to.
   b). Change the JMenuItem text name to reflect your task.
   c). Register the plug-in with the appropriate HEC-DSSVue menu, or you can make a toolbar button and add it to the main toolbar.
   d). Modify the process function(s) to do what you want with the data.
   e). Comment or uncomment code to save changed data with the same pathname or with a different pathname or allow the user to change the pathname, as desired.
   f). File you changes.
4. Compile your code. Add hec.jar, heclib.jar and rma.jar in your class path:
```
javac -classpath "C:\Program Files\Hec\HEC-DSSVue\
jar\hec.jar;C:\Program Files\Hec\HEC-DSSVue\jar
\heclib.jar;C:\Program Files\Hec\HEC-DSSVue\jar\
rma.jar" MyPlugin.java
```
5. Copy the "GenericDssVuePlugin.jar" to a jar file named according to your task in the HEC-DSSVue\Plugins directory.
6. Use pkzip or other zip utility to open your .jar file. Add in the class files that you generated in the compile (maybe more than one). Don't include the directory names, unless you are using

packages, where you will have to use a directory name that matches the package. (Figure B.1).

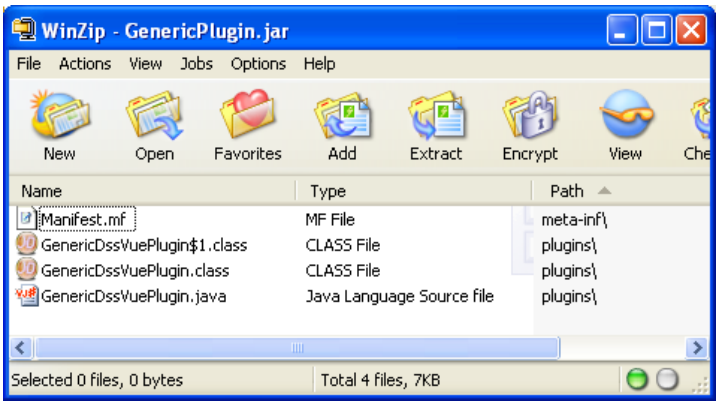

Figure B.1 GenericPlugin.jar Directory

7. Extract the Manifest.mf file with the path. Edit this file with Wordpad and change the GenericDssVuePlugin name to the name of your class. Add the modified file back to the .jar file and be sure that you keep the path specified (meta-inf\). (Figure B.2).

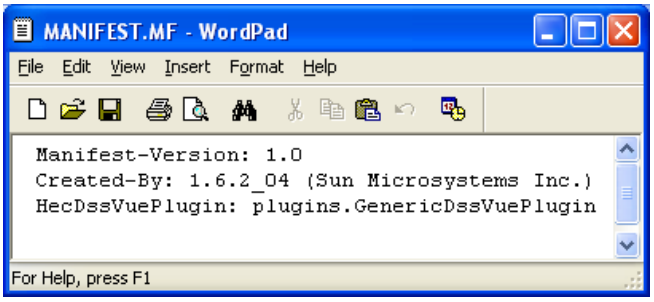

Figure B.2 Manifest *.mf* File

8. You should be good to go at this point. Run HEC-DSSVue and test your plug-in. (Figure B.3). If you have problems, compare what you have to one of the simple example plug-ins. If the run is successful, you will receive a confirmation as is shown in Figure B.4 (page B-4).

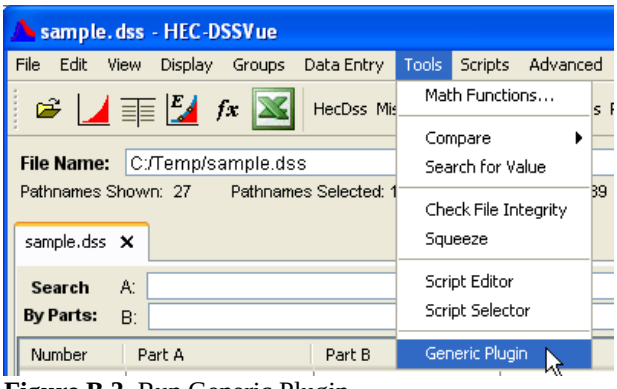

Figure B.3 Run Generic Plugin

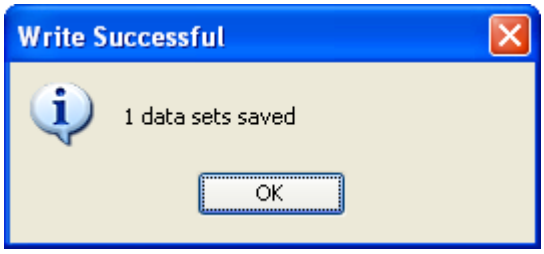

Figure B.4 Successful Run Confirmation

B.4 Generic DssVuePlugin Source

```
package plugins;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JOptionPane;
import java.util.*;
import hec.dssgui.ListSelection;
import hec.io.TimeSeriesContainer;
import hec.heclib.dss.DSSPathname;
import javax.swing.JMenuItem;

public class GenericDssVuePlugin
{
   public static void main(Object[] args)
   {
      final GenericDssVuePlugin plugin = new GenericDssVuePlugin();
      final ListSelection listSelection = (ListSelection) args[0];

      JMenuItem genericMenuItem = new JMenuItem("Generic Plugin");
      genericMenuItem.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e){
            plugin.process(listSelection);
         }
      });
      listSelection.registerPlugin(ListSelection.TOOLS, genericMenuItem);
   }

   protected void process(ListSelection listSelection)
   {
      //  Reads data from DSS and returns it in data containers in a list
of lists
      List[] dataList = listSelection.getSelectedDataContainers();
      if (dataList == null) {
         JOptionPane.showMessageDialog(listSelection, "No records
selected",
                                       "Cannot read data",
JOptionPane.WARNING_MESSAGE);
         return;
      }
      // The first list contains a list of TimeSeriesContainers
      // The second is PairedDataContainers, third is text, fourth is
gridded
      List tsContainers = dataList[0];
      if ( (tsContainers != null) && (tsContainers.size() > 0)) {
         boolean success = processData(listSelection, tsContainers);
         if (!success) {
            JOptionPane.showMessageDialog(listSelection, "Error in
processing data",
                                       "Error", JOptionPane.WARNING_MESSAGE);
         }
      }
      else {
         JOptionPane.showMessageDialog(listSelection, "No time series data
read",
                                       "No data found",
```

```
JOptionPane.WARNING_MESSAGE);
      }
   }

   protected boolean processData(ListSelection listSelection, List
tsContainers)
   {
      for (int i = 0; i < tsContainers.size(); i++) {
         TimeSeriesContainer tsc = (TimeSeriesContainer)
tsContainers.get(i);
         //  Do something with the data here
         //  For this example, we will add "10" to each value
         for (int j=0; j<tsc.values.length; j++) {
            tsc.values[j] += 10.0;
         }
         //  Save the data, if needed
         DSSPathname path = new DSSPathname(tsc.fullName);
         path.setFPart(path.fPart() + " + MOD");
         tsc.fullName = path.pathname();
         //  You can save or saveAs individual containers as you modify
them
         //listSelection.save(tsc);
         //listSelection.saveAs(tsc);
      }
      //  OR you can save the list of containers.
      listSelection.save((Vector)tsContainers);
      //  Update the catalog if you added records
      listSelection.refreshCatalog();
      //  Display a confirmation message
      JOptionPane.showMessageDialog(listSelection, tsContainers.size() + "
data sets saved",
                                    "Write Successful",
JOptionPane.INFORMATION_MESSAGE);
      return true;
   }
}
```

B.5 Primary Functions

B.5.1 ListSelection Functions

To register your plug-in with HEC-DSSVue, the following functions are available:

```
public static void main(Object[] args) {
    final ListSelection listSelection = (ListSelection) args[0];
```

1. Register your plug-in with:
   a. `public void registerPlugin(int menuNumber, JMenuItem menuItem)`

   where *menuNumber* is:

```
FILE = 0;
EDIT = 1;
VIEW = 2;
DISPLAY = 3;
GROUPS = 4;
DATA_ENTRY = 5;
TOOLS = 6;
CUSTOM = 7;
```

```
SCRIPTS = 8;
ADVANCED = 9;
HELP = 10;
```

b. `public void registerImportPlugin(ImportPlugin importPlugin, JMenuItem menuItem, String dropAndDragExtension)`

c. `public void registerExportPlugin(JMenuItem menuItem)`

d. `public void addToolBarButton(JButton button)`

2. You can put your plug-ins in a custom menu that you can name (e.g., listSelection.setCustomMenu(Our Math Functions): `public void setCustomMenu(String menuText)`

3. Read data for selected pathnames with getSelectedData Containers():

```
List[0] will be a list of TimeSeriesContainer, or null if
none read.
List[1] will be a list of PairedDataContainer or null
List[2] will be a list of TextContainer
```

4. Write data to HEC-DSS:

```
boolean save(DataContainer dataContainer);
boolean saveAs(DataContainer dataContainer);
int save(Vector DataContainer);
```

5. Others:

```
DataReferenceSet getSelectedPathnames()
void updateCatalog();
boolean setDSSFilename(String DSSFileName);
String getDSSFilename();
void addToSelection(String pathname)
public void setTimeWindow(String timeWindow)
void setTimeWindow(String start, String end, boolean applyAll)
void setTimeWindow(HecTime start, HecTime end, boolean applyAll)
void tabulate(List selectedDataContainers);
void plot(List selectedDataContainers);
```

B.5.2 DSSPathname

hec.heclib.dss.DSSPathname

Table B.1 DSSPathname

Field	Type	Description
getPathname ();	String	Returns the pathname
getAPart();	String	Gets the A part
getBPart();	String	Gets the B part
getCPart();	String	Gets the C part
getDPart();	String	Gets the D part
getEPart();	String	Gets the E part
getFPart();	String	Gets the F part
setAPart (String aPart);	void	Overrides the default A part
setBPart (String bPart);	void	Overrides the default B part
setCPart (String cPart);	void	Overrides the default C part

Field	Type	Description
setDPart (String dPart);	void	Overrides the default D part
setEPart (String ePart);	void	Overrides the default E part
setFPart (String fPart);	void	Overrides the default F part
setPathname (String pathname);	int	sets the pathname and returns 0 if successful
static getDefaultAPart();	String	Gets the default A part
static getDefaultBPart();	String	Gets the default B part
static getDefaultCPart();	String	Gets the default C part
static getDefaultDPart();	String	Gets the default D part
static getDefaultEPart();	String	Gets the default E part
static getDefaultFPart();	String	Gets the default F part
static setDefaultAPart (String aPart);	void	Sets the default A part
static setDefaultBPart (String bPart);	void	Sets the default B part
static setDefaultCPart (String cPart);	void	Sets the default C part
static setDefaultDPart (String dPart);	void	Sets the default D part
static setDefaultEPart (String ePart);	void	Sets the default E part
static setDefaultFPart (String fPart);	void	Sets the default F part

B.5.3 TimeSeriesContainer

.io.TimeSeriesContainer - contains the data read or to be written:

Table B.2 TimeSeriesContainter

Field	Type	Description
endTime;	int	The end time of the time window. If the data were retrieved, the end time may be later than the last time in the times list.
fullName;	String	Pathname
fileName;	String	Dss filename
interval;	int	The interval, in minutes, between each set of consecutive times in the times list. For irregular-interval times, the interval field is set to -1.
location	String	The location associated with the data (DSS pathname B-part if the DataContainer is associated with the DSS file).
numberValues;	int	The length of values and times lists.
parameter	String	The parameter associated with the data
precision = -1;	int	Digits to the right of the decimal
quality	int[ ]	The optional list of quality flags. If this list is present, there must be a quality for each value
startTime;	int	The start time of the time window. If the data were retrieved, the start time may be earlier than the first time in the times list.
subLocation	String	The sub-location associated with the data.

Field	Type	Description
subParameter	String	The sub-parameter associated with the data.
subVersion	String	The sub-version associated with the data.
times;	int[]	The array of times. There is a one-to-one correspondence between the times array and the values array .
timeZoneID	String	The time-zone for this object. Set to “” if none.
timeZoneRawOffset	int	The offset, in milliseconds, from UTC to the time zone.
type	String	Field containing the data type (PER-AVER, etc)
units	String	The units of the data (CFS, etc)
values	double[]	The data values, each of which has a corresponding time in the times array and optionally a corresponding quality in the quality array. All arrays must have the same length.
version	String	The version associated with the data (DSS pathname F-part if the DataContainer is associated with a DSS file).
watershed	String	The watershed associated with the data (DSS pathname A-part if the DataContainer is associated with a DSS file).

B.5.4 PairedDataContainer

Table B.3 PairedDataContainter

Field	Type	Description
endTime;	int	The end time of the time window. If the data were retrieved, the end time may be later than the last time in the times list.
fullName;	String	Pathname
fileName;	String	Dss filename
interval;	int	The interval, in minutes, between each set of consecutive times in the times list. For irregular-interval times, the interval field is set to -1.
location	String	The location associated with the data (DSS pathname B-part if the DataContainer is associated with the DSS file).
numberValues;	int	The length of values and times lists.
parameter	String	The parameter associated with the data
precision = -1;	int	Digits to the right of the decimal
quality	int[ ]	The optional list of quality flags. If this list is present, there must be a quality for each value
startTime;	int	The start time of the time window. If the data were retrieved, the start time may be earlier than the first time in the times list.
subLocation	String	The sub-location associated with the data.
subParameter	String	The sub-parameter associated with the data.

Field	Type	Description
subVersion	String	The sub-version associated with the data.
times;	int[ ]	The array of times. There is a one-to-one correspondence between the times array and the values array .
timeZoneID	String	The time-zone for this object. Set to “” if none.
timeZoneRawOffset	int	The offset, in milliseconds, from UTC to the time zone.
type	String	Field containing the data type (PER-AVER, etc)
units	String	The units of the data (CFS, etc)
values	double[ ]	The data values, each of which has a corresponding time in the times array and optionally a corresponding quality in the quality array. All arrays must have the same length.
version	String	The version associated with the data (DSS pathname F-part if the DataContainer is associated with a DSS file).
watershed	String	The watershed associated with the data (DSS pathname A-part if the DataContainer is associated with a DSS file).

B.5.5 HecTime

Table B.4 HecTime

Field	Type	Description
endTime;	int	The end time of the time window. If the data were retrieved, the end time may be later than the last time in the times list.
fullName;	String	Pathname
fileName;	String	Dss filename
interval;	int	The interval, in minutes, between each set of consecutive times in the times list. For irregular-interval times, the interval field is set to -1.
location	String	The location associated with the data (DSS pathname B-part if the DataContainer is associated with the DSS file).
numberValues;	int	The length of values and times lists.
parameter	String	The parameter associated with the data
precision = -1;	int	Digits to the right of the decimal
quality	int[ ]	The optional list of quality flags. If this list is present, there must be a quality for each value
startTime;	int	The start time of the time window. If the data were retrieved, the start time may be earlier than the first time in the times list.
subLocation	String	The sub-location associated with the data.
subParameter	String	The sub-parameter associated with the data.
subVersion	String	The sub-version associated with the data.

Field	Type	Description
times;	int[ ]	The array of times. There is a one-to-one correspondence between the times array and the values array.
timeZoneID	String	The time-zone for this object. Set to "" if none.
timeZoneRawOffset	int	The offset, in milliseconds, from UTC to the time zone.
type	String	Field containing the data type (PER-AVER, etc)
units	String	The units of the data (CFS, etc)
values	double[ ]	The data values, each of which has a corresponding time in the times array and optionally a corresponding quality in the quality array. All arrays must have the same length.
version	String	The version associated with the data (DSS pathname F-part if the DataContainer is associated with a DSS file).
watershed	String	The watershed associated with the data (DSS pathname A-part if the DataContainer is associated with a DSS file).

B.5.6 HecDouble

Table B.5 HecDouble

Field	Type	Description
isDefined();	boolean	false if missing or not set
set(double value);	void	Sets the value
set(double value, int precision);	void	Sets the value and precision (the number of values past the decimal place to display)
set(String value);		Sets the value and precision
setPrecision (int precision);	int	Sets the precision – the number of values past the decimal place to display. (does not change what is stored)
string();	String	Returns a String of the value
string(boolean printCommas);	String	Returns a String of the value. If printCommas is true, commas are inserted in the appropriate locations in the string.
string(boolean printCommas, DecimalFormatSymbols symbols);	String	Returns a String of the value, formatted according to the format given.
value();	double	returns the actual number

B.5.7 HecDoubleArray

hec.heclib.util.HecDoubleArray - an array of HecDouble

Table B.6 HecDoubleArray

Method	Returns	Description
element(int elementNumber);	HecDouble	Returns the HecDouble object at that location.
set (double values[]);	void	Sets the values.
setPrecision(int precision);	int	Sets the number of values to display past the decimal point for all values in the array.
string(int elementNumber);	String	Returns the value at the element given as a String
string(int elementNumber, boolean printCommas);	String	Returns the value at the element given as a String. If printCommas is true, commas are inserted at the appropriate locations.

B.6 ToTextPlugin

The ToTextPlugin takes selected time series data sets and writes the data out to a text file. You can use this to create an input file in a certain format for other programs.

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JOptionPane;
import java.util.*;
import java.io.*;

import hec.dssgui.ListSelection;
import hec.io.TimeSeriesContainer;
import hec.heclib.util.HecTime;
import rma.awt.FlatPanelButton;
import javax.swing.JFileChooser;
import hec.heclib.util.HecDoubleArray;
import hec.dataTable.*;

public class ToTextPlugin
{
   public static void main(Object[] args)
   {
      final ToTextPlugin plugin = new ToTextPlugin();
      final ListSelection listSelection = (ListSelection) args[0];

      FlatPanelButton button = new FlatPanelButton("To Text");
      button.addActionListener(new ActionListener()
      {
         public void actionPerformed(ActionEvent e)
         {
               plugin.process(listSelection);
         }
      });
      listSelection.getToolBar().add(button);
   }

    protected void process(ListSelection listSelection)
    {
        //  Reads data from DSS and returns it in data containers in a
list of lists
        List[] dataList = listSelection.getSelectedDataContainers();
        if (dataList == null) {
            JOptionPane.showMessageDialog(listSelection, "No records
selected",
                                          "Cannot read data",
```

```
JOptionPane.WARNING_MESSAGE);
            return;
        }
        // The first list contains a list of TimeSeriesContainers
        // The second is PairedDataContainers, third is text, fourth is
gridded
        List tsContainers = dataList[0];
        if ( (tsContainers != null) && (tsContainers.size()
0)) {
            processData(listSelection, tsContainers);
        }
        else {
            JOptionPane.showMessageDialog(listSelection, "No time series
data read",
                                          "No data found",
JOptionPane.WARNING_MESSAGE);
        }
    }


    protected boolean processData(ListSelection listSelection,
                                  List tsContainers) {
        try {
            //  Get the name of the text file to write to.
            JFileChooser chooser = new JFileChooser();
            chooser.setAcceptAllFileFilterUsed(false);
            chooser.setDialogTitle("Enter file to save data to");
            chooser.setFileFilter(new rma.util.RMAFilenameFilter("txt",
"*.txt"));
            chooser.showOpenDialog(listSelection);
            java.io.File file = chooser.getSelectedFile();
            if (file == null)
                return false;
            String fileName = file.getAbsolutePath();
            if (fileName.endsWith(".txt") == false)
                fileName = fileName + ".txt";
            PrintWriter textOut = new PrintWriter(new
FileWriter(fileName));
            //  Write either as either data in a single column or data
sets
            //  in multi-column.  The table software organizes data sets
with times.
            boolean tableStyle = true;
            if (tableStyle) {
                writeTableFormat(textOut, tsContainers);
            }
            else {
                writeSingleSets(textOut, tsContainers);
            }
            textOut.close();
            JOptionPane.showMessageDialog(listSelection,
                                          tsContainers.size() + " data
sets written to " +
                                             fileName,
                                          "Write Successful",
ptionPane.INFORMATION_MESSAGE);
            return true;
        }
        catch (Exception e) {
            JOptionPane.showMessageDialog(listSelection, e.toString(),
                                                  "Cannot write to file",

               JOptionPane.WARNING_MESSAGE);
            return false;
        }
    }

    protected void writeSingleSets(PrintWriter textOut, List tsContainers)
{
        HecTime time = new HecTime();
        for (int i = 0; i < tsContainers.size(); i++) {
            TimeSeriesContainer tsc = (TimeSeriesContainer)
```

```
tsContainers.get(i);
            textOut.println(tsc.fullName);
            //  HecDouble will take care of missing data and the precision
            HecDoubleArray values = new HecDoubleArray();
            values.set(tsc.values);
            values.setPrecision(tsc.precision);
            for (int j = 0; j < tsc.numberValues; j++) {
                time.set(tsc.times[j]);
                textOut.println(time.toString(4) + ";  " +
values.element(j));
            }
        }
    }

    protected void writeTableFormat(PrintWriter textOut, List
tsContainers) {
        HecDataTable dataTable = new HecDataTable(null);
        dataTable.setData(tsContainers, false, 0);
        // UndefinedStyle: 0 = ""; 1 = -901.0; 2 = "M"; 3 = "-M-"
        dataTable.setUndefinedStyle(2);
        //  See HecTime (or heclib juldat) for date styles
        dataTable.setDateStyle(4);
        int numberColumns = dataTable.getColumnCount();
        int numberRows = dataTable.getRowCount();
        int startingRow = dataTable.getNumberHeaderRows();
        for (int i=startingRow; i<numberRows; i++) {
            StringBuffer sb = new StringBuffer();
            for (int j=0; j<numberColumns; j++) {
                sb.append((dataTable.getValueAt(i, j)).toString());
                sb.append("   ");
            }
            textOut.println(sb.toString());
        }
    }
}
```