What is the DataViews Plug-In?

The DataViews Netscape Navigator Plug-In is a Web extension to DataViews that lets you display and animate DataViews views in a web browser. With this capability, you can access real-time data from anywhere you use Netscape Navigator. You can use the same views you created for your DataViews applications simply by saving them in ASCII with a different extension. You can also establish links to other views or web pages simply by using a URL specification in your object names.

To invoke the DataViews Plug-In, you can display a view directly in your browser or display a web page that contains a reference to a view. When the Plug-In is active, you see the view and can use input objects, jump to other views, or display other web pages by clicking on objects in the view. When the view is contained within a web page, you have control over the size of the rectangle containing the view, you can activate the dynamics in the view, and you can set up a data connection through CGI scripts.

User interaction can be handled directly by the view (through input objects) or through three types of CGI (Common Gateway Interface) scripts on the server. These types are:

- data source scripts
- data initialization scripts
- data sink scripts

Data source scripts act as a live data feed to a Plug-In instance. Data initialization scripts initialize the Plug-In instance by feeding data to it before the view is displayed. Data sink scripts let you react to user input such as button clicks or mouse movement. Using these CGI scripts, you can program a web session much as you program a DV-Tools application.

To use the Plug-In, all you have to do is:

- Install the Plug-In.
- Save your DataViews Views in ASCII with an appropriate file extension.
- Register the DataViews MIME type with your web server.
- Provide a data source script, if desired.
- Provide a data initialization script, if desired.
- Provide a data sink script, if desired.
- Embed the view in an HTML page or display it directly in the web browser.

This document describes all of these steps.

Requirements

To run the DataViews Plug-In, you must have the following components:

- DataView 9.8
- Netscape Navigator 3.0 or higher

Installing the Plug-In

The Plug-In is installed when you install DataViews 9.8 unless you do a custom installation and elect not to install it.

If you did not install the Plug-In when you installed the rest of the DataViews product, you can install the Plug-In separately by re-running the DataViews Setup Program.

To verify that the Plug-In is installed, run Netscape and select *About Plug-Ins* from the *Help* menu. The Plug-In list should include the following information: *DataViews 9.8 for Windows Netscape Plug-In v 1.0*.

Using the Plug-In

The Plug-In displays your DataViews views in a client-server relationship. The client is your web browser. This can be located on any machine that has access to the Internet or your local Intranet. The server is where your CGI data source, data initialization, and data sink scripts are located. Text file data source and data initialization scripts as well as views can be located on any convenient machine. This relationship lets you monitor real-time data from almost anywhere.

The following figure illustrates this relationship:

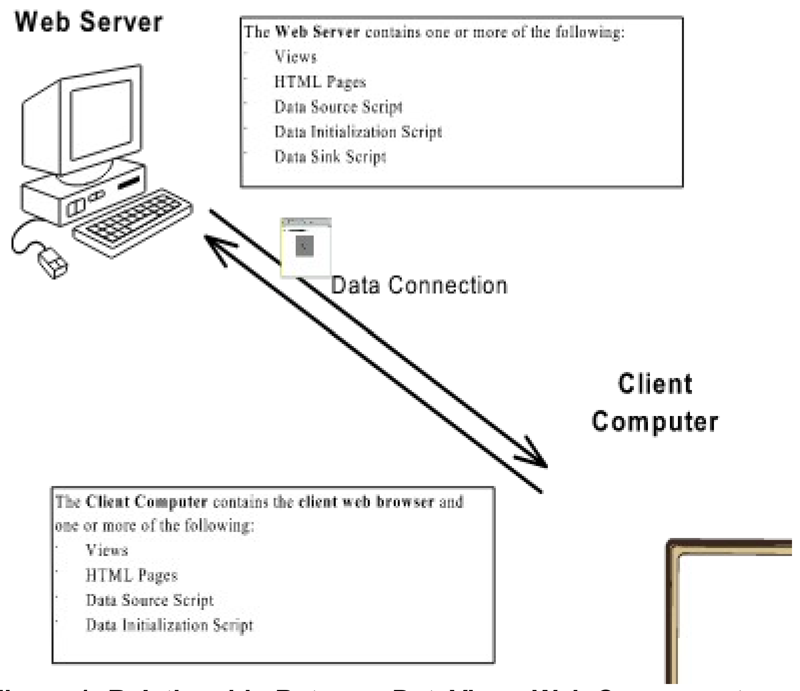

Web Server
The Web Server contains one or more of the following:
Views
HTML Pages
Data Source Script
Data Initialization Script
Data Sink Script
Data Connection
Client Computer
The Client Computer contains the client web browser and one or more of the following:
Views
HTML Pages
Data Source Script
Data Initialization Script

Figure 1: Relationship Between DataViews Web Components

Building DataViews Views

You build a view with DV-Draw. You can display any DataViews view using the Plug-In. All you need to do is:

- Save the view in ASCII mode instead of Binary.

  Using ASCII view files is critical in a cross-platform environment. Web servers run on many platforms, as does Netscape Navigator.

  You can set the DV-Draw save mode by choosing *View->preferences*, then clicking the ASCII button in the save mode area. You can also use the utility *viewconv*, located in the *<DVHOME>\bin* directory, to convert views from binary to ASCII representation.

- Save the view with a *.dvv* or *.dvweb* extension.

  The Plug-In registers *.dvv* and *.dvweb* to the DataViews MIME type. If the view is on a local file system (not on a web server) these extensions are the only way to invoke the Plug-In. In this case, displaying a view with any other extension simply shows the ASCII representation of the view in the browser window.

  This restriction applies only to the top-level views. Referenced views can have any extension.

  There is no difference between the *.dvv* and *.dvweb* extensions. Both are registered to and invoke the Plug-In.

  If the view is located on a web server, the server can be configured so that a top-level view with any extension, including *.v*, starts the Plug-In. See the Registering the DataViews MIME Type with your Web Server section, later in this manual.

Creating Web Links to Other Views, Web Pages, or Graphics Files

The name of an object can be used to establish a link to other views, *.gif* files or web pages by formatting it as a URL. Clicking on the object traverses the link to the designated URL which is then displayed by Netscape. The format for the object name that makes it act like a link when displayed by the Plug-In is:

<href="*referenced_file"*>

where *referenced_file* is either a valid web address that starts with http:// or a file in the local directory hierarchy:

<href="test.dvv">	local file
<href="cool.gif">	local file
<href ="http://www.dvcorp.com">	web server reference
<href="http://www.dvcorp.com/mp/dv/ DataViews.html>	web server reference

The *DV-Draw User's Guide* gives complete instructions for creating views. This document assumes you know how to build a view using the default data source, memory variables, and input objects. If any of these terms is unfamiliar to you, please see the relevant sections in the *DV-Draw User's Guide*.

Registering the DataViews MIME Type with Your Web Server

Before you can display DataViews views with the Plug-In, you must set your web server to recognize that it is sending a view to a client browser. While your server already recognizes such file types as: *.gif*, *.jpeg*, and *.html*, you have to add the DataViews view type to the server setup. The server recognizes files types by their extension. Each extension is associated with a corresponding MIME type. A MIME type is a textual representation of the type of information that is being sent to or from a web server. For example, the MIME type for *.gif* files is *image/gif*, for *.jpeg* files is *image/jpeg* and for *.html* files is *text/html*. You can associate more than one file extension with the same MIME type. This association is usually made in a configuration file on the server.

The MIME type for DataViews views is:

```
text/x-dvweb
```

You can set up your web server to associate *text/x-dvweb* with file extensions such as *.dvv*, *.dvweb* and *.v*. Consult your Web Server documentation to determine how to set up these associations. Appendix C: Setup for Common Web Servers also contains setup instructions for some of the more common Web Servers.

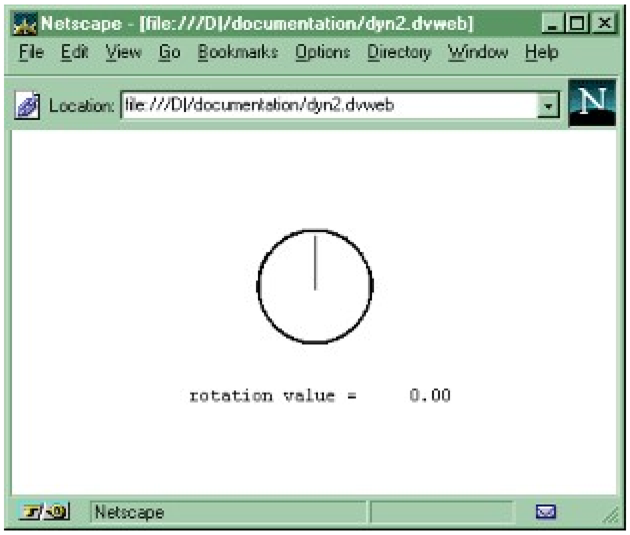
After you do this, files with names such as *myview.dvv*, *myview.dvweb*, and *myview.v* that are located on a web server all generate the correct HTML header, which gets sent to the browser. The browser, which is set up to load the DataViews Plug-In for a MIME type of *text/x-dvweb*, starts a Plug-In instance for each of these files. A header supplied by a web server supersedes the need for the browser to check the file's extension. Therefore, your web server can overrule the Plug-In's file extension requirements of *.dvv* or *.dvweb*. The Plug-In never checks the file extension when a MIME header is supplied.

Loading a View Directly into the Browser

The quickest way to display a view with the Plug-In is to load the view directly as a file.

Select *File..Open File* in Netscape Navigator or *File..Open File in Browser* in Nestcape Navigator Gold. Browse to the correct directory and double click on the name of the DataViews view file that you'd like to see. Remember that the view file must have a *.dvv* or *.dvweb* file extension.

You should now see the view in the browser window:


Netscape - [file:///D|/documentation/dyn2.dvweb]
File Edit View Go Bookmarks Options Directory Window Help
Location: file:///D|/documentation/dyn2.dvweb
Netscape

rotation value = 0.00

Loading a view this way is mainly good for testing that your setup works. Notice that it is stretched to fit the entire area of the browser so objects may be distorted. You have no control over the height and width of the display area for the view. Input objects are active, but you cannot start a built-in data source. Also, you cannot use many of the advanced Plug-In features such as using a web server based data source or programmable reaction to user feedback.

If you have trouble displaying views in your browser, see Appendix B: Troubleshooting/ FAQ .

Embedding a View in a Web Document

To control the size of the rectangle in which the Plug-In is displayed, to connect data to the Plug-In, and to use other HTML along with the Plug-In, you embed the Plug-In in a Hypertext Markup Language (HTML) document.

To display an embedded view, you use the HTML <*EMBED*> command. <*EMBED*> has various attributes that you set to let the browser know what is being embedded and how it should appear. Here is an example of an <*EMBED*> command that displays a DataViews view:

```
<EMBED SRC="http://ithica/dyntest.dvweb"
WIDTH=600 HEIGHT=600
NAME=MY OBJECT>
```

This command uses the *SRC*, *WIDTH*, and *HEIGHT* attributes. These attributes are part of the standard set of attributes for any embedded object. The attribute NAME is specific to the DataViews Plug-In.

The SRC attribute specifies the top-level view to load into the Plug-In. The *WIDTH* and *HEIGHT* attributes control the width and height of the rectangle displaying your view. The *NAME* attribute assigns a name to the Plug-In instance so that a server program can send commands specifically to that instance. Sending commands to the Plug-In is discussed in the Using Live Data: The External Data Source section and the Using Live Data- Interactions with the User via the Data Sink section. You can also add other HTML tags besides <*EMBED*> before and after the view.

The following example shows an HTML document with an embedded view. Note that several HTML tags are used in addition to the <*EMBED*> command:

```
<HTML><HEAD><TITLE>My Home Page</TITLE></HEAD>
<BODY>
<H1>Embedded view in HTML document</H1>
<center><p>This is an example of a DataViews view embedded in a web page.
Note that you can use any HTML tags, not just the &ltEMBED&gt tag.</p>
<EMBED SRC="http://ithica/dyntest.dvweb"
WIDTH=600 HEIGHT=600
NAME=MY OBJECT></center>
</Body></HTML>
```

This file, saved as *home.htm* and loaded into the web browser, looks like this:

Additional attributes are available for the DataViews Plug-In. These let you further customize the representation of the Plug-In in the browser. Some of these other attributes are discussed later in this document. The full set of attributes is contained in Appendix A: DataViews PlugIn Attributes .

Animating the View

If you have set up your view with a data source, you can animate the view on the web easily. You use two *<EMBED>* attributes that are specific to the DataViews Plug-In. These are *AUTOREADDATA* and *OPENDATASOURCES*:

- **AUTOREADDATA**

  If set to *TRUE*, every time the Plug-In automatically updates, it also performs a *TviReadData()*. The default is *FALSE*. Typically, this is only important if the view uses file or function data sources.

- **OPENDATASOURCES**

  If set to *TRUE*, view data sources are opened before the initial draw. The default is *FALSE*. Typically, this is only important if the view uses file, function, or constant data sources.

This is an example of the *<EMBED>* code you need to animate a view with a data source:

```
<EMBED
SRC="http://ithica/dyntest.dvweb"
AUTOREADDATA=TRUE
OPENDATASOURCES=TRUE
WIDTH=600 HEIGHT=600
NAME=MY OBJECT
>
```

When you load an HTML file with this *<EMBED>* code, the Plug-In displays the *dyntest.dvweb* view, animated by its internal data source.

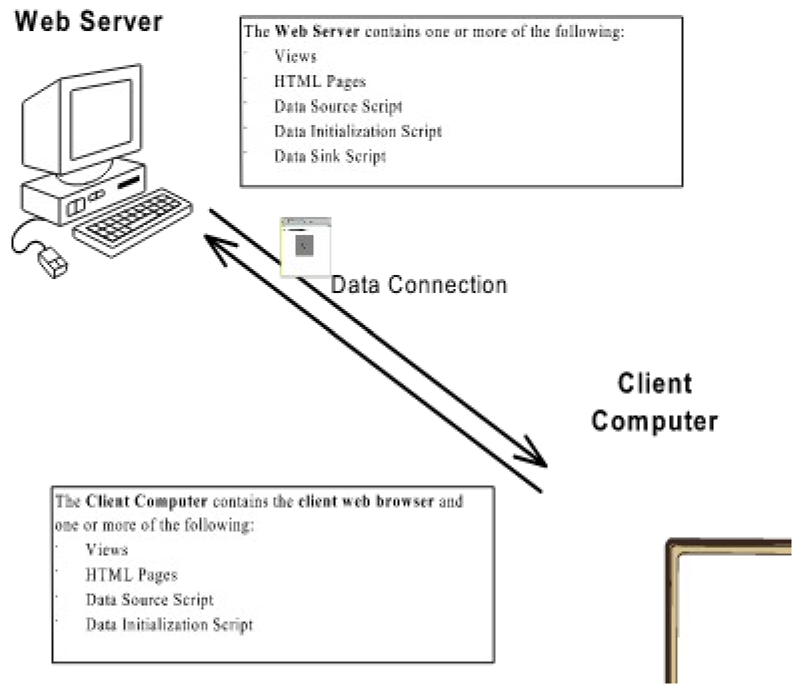
In later sections, we discuss other *<EMBED>* attributes and more advanced procedures such as attaching a server-based data source to your view and attaching a server-based data sink to your view. These procedures allow your view to react to user interaction.

The rest of this document explains the DataViews Plug-In Client/Server model, how to embed references to DataViews views in an HTML document, how to establish and hook up an external data source to the view, and how to react to user input.

Using Input Objects

One way to simulate live data is to use Input Objects in your view. Attach the input object to a memory variable that controls one or more dynamics on an object.

When you display the view in the web browser, you can use the input object to change the memory variable. If the memory variable is attached to a line's rotation dynamics, for example, changing the input object rotates the line. Here, the *<EMBED>* command does not need the *OPENDATASOURCES* and *AUTOREADDATA* commands because we are not relying on the default file data source.



The **Web Server** contains one or more of the following:
- Views
- HTML Pages
- Data Source Script
- Data Initialization Script
- Data Sink Script

The **Client Computer** contains the **client web browser** and one or more of the following:
- Views
- HTML Pages
- Data Source Script
- Data Initialization Script

Figure 1: Relationship Between DataViews Web Components

The DataViews PlugIn Client/Server Model

In this section, we replace the input object mentioned in the previous section with a live data source on the web server. The data source sends commands to the Plug-In to change the memory variable attached to the line's rotation dynamics.

The previous examples do not discuss the web server, the data source script, the data initialization script, and the data sink scripts that appear in Figure 1 . This is because the previous examples were fairly simple. For example, when you point your browser directly to a view file, that file can be located on a web server or on your local machine. The same is true when you embed a view in a web document but do not attach an external data source or data sink. In either of these cases, you do not get any "extra" benefit from using the server besides the ability to look at a view located on some other machine.

The real power of the Plug-In is that it allows you to serve data to a view and react to user interaction across the Internet or your local Intranet. This power is derived from small "glue" programs called Common Gateway Interface (CGI) scripts located on a web server, such as the data source, data initialization, and data sink scripts mentioned earlier.

In this model, the web document may or may not be located on a web server. The view may or may not be located on a web server. If the data source or data initalization script are plain text files, they can be located anywhere However, CGI scripts (data source, data initialization, or data sink) must be located on the web server.

When you start a Plug-In instance, the following events occur:

1. If a *DATAINIT* attribute is defined, the Plug-In runs that program before it displays the view. The *DATAINIT* program sends commands to the Plug-In to initialize the view. The *DATAINIT* is run once at the start of a Plug-In instance and must exit before anything else happens.
2. The view is displayed.
3. If a *DATASOURCE* attribute is defined, the Plug-In starts the *DATASOURCE*. This program runs constantly, sending commands to the view until the Plug-In instance is destroyed.
4. If a *DATASINK* attribute is defined, query strings are sent to it when the user interacts with something in the view. A query string is a text string that defines the user interaction. The *DATASINK* responds to the query string with commands sent to the view. The *DATASINK* program is started each time a user interacts with the view. The *DATASINK* program must return commands then exit before anything else happens.

Before you can attach a data source CGI program to your view and react to user input with a data sink CGI program, you need to be sure that your web server is set up to handle the DataViews Plug-In. See the Registering the DataViews MIME Type with Your Web Server section earlier in this document.

Using Live Data: The External Data Source

In this section, we discuss how to feed real-time data to your views from a web server. If you have programmed external data sources for DV-Tools applications, this process should be familiar to you. You should know how to use a scripting or batch language to create an external data source and you must have permission to access your web server.

An external data source is an executable program or a text file that feeds commands to your view through the Plug-In. These commands are text-based and are fed to the Plug-In through *stdout* by your web server.

The Plug-In attribute DATASOURCE connects your data source CGI script or text file to your view. The *An Example Web Page* section, later in this document, contains an example of using the *DATASOURCE* attribute.

The following commands can be used by the *DATASOURCE*, *DATAINIT*, and *DATASINK* scripts (The *DATAINIT* and *DATASINK* attributes are discussed later in this document):

- **alias** *name value*

  This command sets an alias *name* for the long string *value*. You use an alias by putting an '@' in front of the *name*:

  The following example commands to a Plug-In instance set up aliases:

  ```
  alias s1   set_value default.mem/var:1 f 1.0
  alias s    set_value default.mem/var:1 f
  ```

  The Plug-In instance can then use these aliases to send much shorter commands. The following lines:

  ```
  @s1
  @s 2.0
  ```

  are interpreted as:

  ```
  set_value default.mem/var:1 f 1.0
  set_value default.mem/var:1 f 2.0
  ```

  Each Plug-In instance maintains its own list of aliases.

- **draw_next**

  This causes a TdpDrawNext() to occur on the view.

- **pause**

  This makes the Plug-In suspend periodic updating of the display until a *resume* command is received. If large amounts of data are coming across the net, you can use *pause* and *resume* to keep the graphics from updating until all the data has been received.

  pause has no effect when periodic updating is off. To start periodic updating, send the *period* command to the Plug-In with a non-zero period (see the *period* explanation below).

- **period** *milliseconds*

  Sets the period of automatic updating, measured in milliseconds. If milliseconds is 0, the Plug-In no longer automatically updates.

- **resume**

  Resumes periodic updating using the previous *period*. See *pause*.

- **set_value** *ds/dsv value*

Sets the named view data source variable to the given value. *ds/dsv* stands *for <datasource_name>/<datasource_variable_name>*.

- **tell** *name command*

  Directs a command to a particular named Plug-In instance. For example,

  ```
  tell IndexView set_value default.dat/Var:1 f 1.0
  ```

  sets the default data source variable in the top-level view contained in the *IndexView* Plug-In instance to 1.0.

- **update**

  Causes a TviReadData() to occur, updating any history buffers in the view. See the *DV-Tools Reference Manual* for a complete explanation of *TviReadData()*.

Abbreviating Commands

You can abbreviate all of the commands to one or more initial letters except **period**, which requires at least "pe" to differentiate it from **pause**.

An Example Web Page

The following example shows a web page that uses a data source with a view:

```
<HTML>
<H1> DataViews Sample Plug-In </H1>
<HR>
<p><center>
<EMBED SRC="http://Ithica/dynamics.dvweb"
SERVERPATH="views"
DATASOURCE=http://Ithica/dyn_data.x
TYPE="text/x-dvweb"
ALWAYSREQUESTFILE=TRUE
WIDTH=600
HEIGHT=600
NAME="MY OBJECT">
<HR></HTML>
```

Three new attributes are used in this example:

- **SERVERPATH**

  This is a path to search on the server.

- **ALWAYSREQUESTFILE**

  If set to *TRUE*, any embedded pixmap, subdrawing, and file data source files are always requested from the server even if a local copy is available. If it is set to *FALSE* (the default), and if the file was found locally (by using the ordinary DV-Tools search paths), the local copy is used instead.

- **TYPE**

  This is an attribute that is common to all *<EMBED>* objects. This lets you specify the MIME type of the embedded object rather than letting the server send this information.

Sending PlugIn Commands across the Network

As stated earlier, Plug-In commands are sent via *stdout* to the Plug-In which acts on the view. But what does this mean? How do the commands get to *stdout* and how does the Plug-In know about them and know where to send them? We answer these questions in this section.

The first step to using an external data source is to set aside one or more memory variables in your view for the live data. For details about setting up memory variables, see the *DV-Draw User's Guide*.

The next step is to design your data source. This can be, as in the simplest case above, a text file with a list of Plug-In commands. The *<DVHOME>\plugin\exampes\sinwave\* directory contains an example of a text file data source. In a more complex and useful case, it can be a CGI-BIN script written in any language that can write to *stdout.* CGI-BIN scripts must be located on the web server.

In either case, you connect your data source to your view with the *DATASOURCE* attribute of the *<EMBED>* command. For a discussion of the *<EMBED>* command, see Embedding a View in a Web Document earlier in this document. The following example is a modified version of the example in the An Example Web Page section:

```
<EMBED
SRC="http://Ithica/dyntest.dvweb"
AUTOREADDATA=FALSE
OPENDATASOURCES=FALSE
DATASOURCE=”http://Ithica/cgi-bin/dynsource.x”
WIDTH=600 HEIGHT=600 NAME="MY OBJECT"
>
```

Specifying the data source this way starts an instance of that data source on the web server. The data source then feeds Plug-In commands to the view. In the case of a text file, these commands run out after a period of time. With a script, you can program an infinite loop that constantly feeds data to your view. You can use an infinite loop because the Plug-In starts the data source when the instance of the Plug-In is created and destroys the data source when the user moves away from the web page containing the DataViews view.

This is an example of a script written in the C language:

```
#include <math.h>
#include <stdio.h>

int main (argc, argv)
    int argc;
    char *argv[];
{
  float dial_input = 0.0, text_input = 0.0;
  float low_range_out = 170.0, high_range_out = -170.0;
  double low_range_in = -2.0, high_range_in = 2.0;
  int n;

  /* Necessary to indicate to Netscape type of docuement information */
  printf("Content-type: text/plain\n");
  printf("\n");

  /* Loop: change the dial input,text input and update the dynamic objects */
  n=0;
  for(;;)
   {
   if( n == 2000 ) n = 0;

   dial_input = sin ((double) (n / 15.0)) - cos ((double) (n / 25.0));
```

```
    /* Get a mapped output value for the angle */
    text_input = dial_input * (high_range_out - low_range_out) /
        (high_range_in - low_range_in);

    printf( "set_value default.mem/dial_input f %f\n", dial_input );
    printf( "set_value default.mem/text_input f %f\n", text_input );
    printf( "draw_next\n" );
    fflush(stdout);
    n++;
    sleep(.8);
    }
  return 0;
}
```

This program, when compiled, is called *dyn_data.x*. As the name implies, it supplies dynamic data to a view. You can see that all communication with the view is done through *STDOUT* (*printf* statements) and that the dynamic data is supplied through an infinite loop, (`for (;;)`).

In Note 1, you see a MIME header. In this case, it is *text/plain*. Recall that the plain text file example also contained a header. MIME-type headers are important when communicating between a web server and web client.

In the loop, some calculations are made, then Plug-In commands are generated and sent to *STDOUT* by the code in Note 2. This code assumes two memory variables: *dial_input* and *text_input*. Note that we flush the output buffer to be sure the commands are sent. Also, a pause is inserted so that the Plug-In instance does not overwhelm the web browser with data updates.

The DATAINIT Attribute

The *DATAINIT* attribute acts much like the *DATASOURCE* attribute but with the following important differences:

- The commands in *DATAINIT* are only read once, whereas commands in *DATASOURCE* can be read periodically .
- The drawing is not displayed until after the *DATAINIT* stream completes. If there is no *DATAINIT* attribute, the drawing is displayed before the *DATASOURCE* is started.

The *DATAINIT* can be a text file with Plug-In commands or a CGI program, just like the *DATASOURCE*.

An advanced use of *DATAINIT* is to restore historical data to a Plug-In instance when a user goes away then returns to a web page containing a DataViews view. When a user moves to a different web page, the current Plug-In instance is destroyed. If that instance has a *DATASINK* attribute, then the *DATASINK* program can recognize the destruction and create a new *DATAINIT* containing historical data. Then, when the users goes back to the page, the view is passed historical data through the new *DATAINIT* before the *DATASOURCE* is started. This lets it appear as if moving away from a page does not affect the data being displayed.

Using Live Data- Interactions with the User Via the Data Sink

The *DATASINK* attribute specifies a CGI program that receives and responds to user input. The user input is communicated to the CGI program via a query string generated by the Plug-In and sent to the program through *stdin*. The query string includes the name, type, and values from the object with which the user interacted. After parsing the query string to determine which object originated the input and examining the values, the data sink CGI sends commands back to the sending Plug-In instance or to other named instances to modify appearance and behavior.

The commands sent back to the Plug-In are the same as the ones used in the *DATASOURCE* and *DATAINIT* programs. However, the *DATASINK* program is started each time the user interacts with a view. The Plug-In then waits for command output from the *DATASINK* before updating the view. Therefore, the function of the *DATASINK* is to receive commands, react to commands, and exit quickly. This is different from the *DATASOURCE* command. The *DATASOURCE* is started only once, at Plug-In instance creation, and is only killed when the instance is killed.

The following diagram illustrates the *DATASINK*'s interaction with a view:

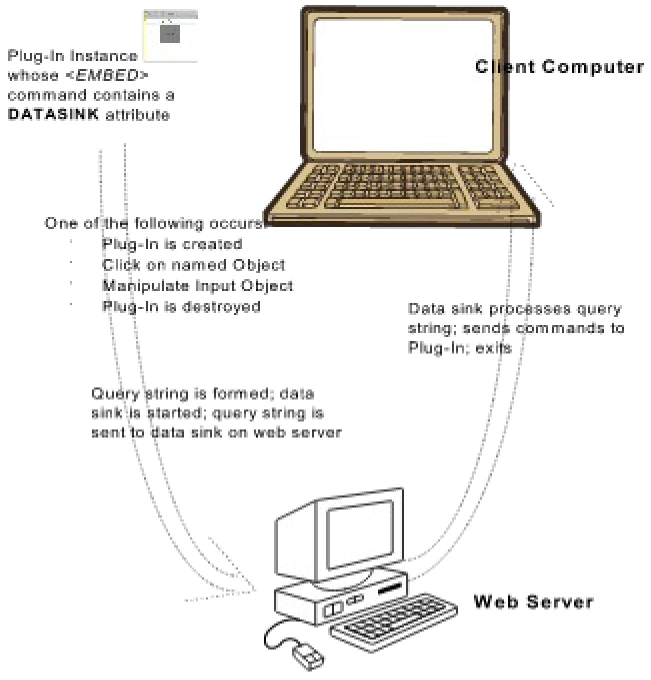

Plug-In Instance whose <EMBED> command contains a DATASINK attribute
Client Computer
One of the following occurs:
Plug-In is created
Click on named Object
Manipulate Input Object
Plug-In is destroyed
Data sink processes query string; sends commands to Plug-In; exits
Query string is formed; data sink is started; query string is sent to data sink on web server
Web Server

Figure 2: Data Flow When using a Data Sink

When a query string is created by the Plug-In, an environment variable, *CONTENT_LENGTH*, is loaded with the length of the query string. Then the *DATASINK* is started and the query string is sent to the DATASINK via *stdin*. The DATASINK must allocate a buffer based on the length of the incoming query string then act on the information given.

The *DATASINK* is notified about two types of user interaction: a user click on a named object, or user manipulation of a named input object. In addition, the *DATASINK* is notified when a Plug-In instance is created and when it is destroyed. The *&Callback* line of the query string indicates the type of notification the *DATASINK* receives:

`&Callback=select`	The user clicks on an object
`&Callback=input`	The user manipulates an input object
`&Callback=create`	The Plug-In instance connected to the *DATASINK* is created
`&Callback=destroy`	The Plug-In instance connected to the *DATASINK* is terminated

The following three sections explain the format of the query string for each type of input notification.

The User Clicks on an Object in a View

The User Manipulates an Input Object

The Plug-In Instance is Created or Destroyed

The User Clicks on an Object in a View

If you left-click on a named object, the Plug-In generates a query string that looks like this:

```
Src=http://ithaca:8080/views/top.dvweb
&Name=My+View
&Id=0x11212312
&Callback=select
&ObjectName=name1
&Button=Left
```

The *Name* field in the above example is the Plug-In instance name, which is set by the *NAME* attribute of the *<EMBED>* command. The *ObjectName* field is the name of the object within the view that the user selected. The *DATASINK* is not notified of clicks on unnamed objects.

Notice that the space in the instance name is replaced with a "+". This means that the text is “URL encoded.” There are serveral rules for URL encoding a text string. For more information, see your server documentation. There are also several freely available packages for parsing information that is sent to a CGI program this way. These libraries make working with CGI input and output much easier.

The User Manipulates an Input Object

Input notifications are more complicated than click notifications, because there is a variety of different input object types each sending different values. Common to all of these are a object name field (as in the click case, above) and an action field that describes what action the user performed. The action is *INPUT_ACCEPT*, *INPUT_DONE*, or *INPUT_CANCEL*. These actions correspond to the definitions given in the DV-Tools manuals for input objects.

After the action comes the new value(s) for the input object. The value is always preceded by an identifier of the form *Valuen*= where n is a value from 1 to the number of values associated with this type of input object. The following examples show the new value commands for various kinds of input objects.

The new value from a Text Editor input object is a single URL-encoded string. For example:

```
Src=http://ithaca:8080/views/top.dvweb
&Name=My+View
&Id=0x11212312
&Callback=input
&ObjectName=name1
&Action=INPUT_DONE
&Value1=New+Text+Here
```

New values from a 2D-Slider are two floats. For example:

```
...
&Value1=0.5
&Value2=0.78
```

New values from a Checklist are expressed as a float for each element of the Checklist. For example:

```
...
&Value1=0.0
&Value2=1.0
&Value3=0.0
.
.
.
```

The new value from a Button input object, Slider, or Toggle, is a single float. For example:

```
...
&Value1=0.0
```

The Plug-In Instance is Created or Destroyed

A query string contains a *Callback* value of *create* when the instance is first created, and a *Callback* value of *destroy* when the instance is destroyed.

This is an example of a query string resulting from instance creation:

```
Src=http://ithaca:8080/views/top.dvweb
&Name=My+View
&Id=0x11212312
&Callback=create
```

This is an example of a query string resulting from instance destruction:

```
Src=http://ithaca:8080/views/top.dvweb
&Name=My+View
&Id=0x11212312
&Callback=destroy
```

As stated in the *DATAINIT Attribute* section, when a *destroy* is sent to the data sink, you can write out a data initialization text file with commands to restore the Plug-In to its previous state. This way, if the user moves away from a page containing a DataViews View then goes back to it, the view appears to pick up where it left off rather than restarting.

See the Data Sink example in the *<DVHOME>\plugin\examples\automobile\* directory for a working example. Note that to run the example, you must transfer and compile the *autosink.c* data sink program to the CGI directory on your server. For detailed information, see the *README* file in the *<DVHOME>\plugin\examples\automobile\* directory.

Debugging Your Data Sink

Normally, the text commands from your data sink get fed directly to the Plug-In instance that called the data sink. This makes it difficult to see whether the output is what you intended. To debug data sink output, you can use the TARGET attribute of the *<EMBED>* command. This attribute, when defined, redirects data sink output to a named window or frame. This allows you to see the output from your data sink program.

Typical TARGET statements are:

`TARGET="_current"`	Displays data sink output in the current Netscape window.
`TARGET="_blank"`	Displays data sink output in a new Netscape window.

In addition to this attribute, most web servers contain a utility data sink whose output is the query string that was sent to it. For NCSA servers, this utility is called *post-query*. You can easily create your own if your server does not supply one. So, to see the query strings being sent to your data sink, you can set the *TARGET* to a named window, such as *_blank*, and set the *DATASINK* attribute to the *post_query* program.

Appendix A: DataViews PlugIn Attributes

The Plug-In, through the HTML *<EMBED>* command, understands the following attributes. Note that the only required attributes are *SRC*, *WIDTH*, and *HEIGHT*.

Properly formed HTML strings have double quotation marks around them. So, for attributes that require a string argument, uses double quotes. This restriction does not apply to *TRUE*/*FALSE* values since these are considered Boolean values, not string values. For example:

`DATASINK="http://myserver/dsink.exe"`	Properly formatted string value.
`ALWAYSREQUESTFILE=TRUE`	Properly formatted Boolean value.

- **ADDTODVPATH**

  This is a semicolon-delimited list of paths to append to the DV-Tools search path. If a file that the view needs is supposed to be found on the client rather than on the server, you may need to set this attribute to ensure that the Plug-In looks in the appropriate directories. Note that all Plug-In instances share the same *DVPATH* information, so changing this value may affect other Plug-Ins.

- **ALWAYSREQUESTFILE**

  If set to *TRUE*, any embedded pixmap, subdrawing, and file data source files are always requested from the server even if a local copy is available. If it is set to *FALSE* (the default), and if the file was found locally (by using the ordinary DV-Tools search paths), the local copy is used instead.

- **AUTOREADDATA**

  If set to *TRUE*, every time the Plug-In automatically updates, it also performs a TviReadData() . The default is *FALSE*. Must be set to *TRUE* for the view to use function data sources.

- **DATAINIT**

  Initializes the view displayed by the Plug-In instance. *DATAINIT* commands are the same as DATASOURCE commands. However, all commands are run before the view is displayed.

- **DATASINK**

  Specifies a CGI program to receive user input from controls or objects in the Plug-In drawing and acts on that input.

- **DATASOURCE**

  This is the URL of a source of animation commands. Animation commands are generally contained in a CGI program on the server but can be supplied in a text file.

- **DEBUGTRACE**

  Turns on debug messages if *TRUE*. The default is *FALSE*. Saves messages to the file associated with DEBUGTRACEFILE .

  You can use the environment variable *DV_DEBUGTRACE* to globally turn on the debug trace. This might be easier than adding the attribute to each instance you want to debug.

  See also: DEBUGTRACEFILE

- **DEBUGTRACEFILE**

  File to save DEBUGTRACE information. Note that this file grows rapidly. Also, if this attribute is set, but *DEBUGTRACE* is *FALSE*, a zero length *DEBUGTRACEFILE* is created.

  You can use the environment variable *DV_DEBUGTRACEFILE* to globally set the debug trace file. This might be easier than adding the attribute to each instance you want to debug.

See also: DEBUGTRACE

- **DISPLAYDVCOPYRIGHT**

  Determines whether to display or suppress the DataViews copyright screen that ordinarily appears before a view is loaded. The default is *TRUE*.

- **ENABLEINPUTCALLBACKS**

  If set to *TRUE* (the default), and if a data sink has been provided, the data sink is notified whenever the user manipulates a *named* input object. The data sink is informed of the name of the input object, the action performed (*DONE*, *CANCEL*, *ACCEPT*, etc.) and the value(s) of the input object's vdp(s).

- **ENABLESELECTIONCALLBACKS**

  If set to *TRUE* (the default), and if a data sink has been provided, the data sink is notified whenever the user clicks on a *named* object. An exception is if the name looks like an HREF, in which case that HREF is requested from the net. The data sink is notified of the name of the object clicked on as well as the mouse button used.

- **HEIGHT/WIDTH**

  The dimensions of the instance if it is embedded into an HTML page. These can be either an integer, meaning the size is that number of pixels, or an integer followed by a percent sign "%", meaning the dimension is that percent of the Navigator window. The SRC , *HEIGHT* and *WIDTH* attributes are the only required attributes.

- **INITIALLYPAUSED**

  If this is set to *TRUE*, the Plug-In does not start automatic updating until it receives an explicit *resume* command from the DATASOURCE . The default is *FALSE*.

- **INITIALLYVISIBLE**

  If this is set to *TRUE*, the top level view is visible as soon as it has been loaded, regardless of whether there is a DATAINIT stream. If set to *FALSE*, the view is hidden until the *DATAINIT* is complete.

- **NAME**

  Names the Plug-In instance. The value of this attribute is used in conjunction with the tell Plug-In command. When an instance is named, commands can be sent to it from other instances.

- **OPENDATASOURCES**

  If set to *TRUE*, view data sources are opened before the initial draw. The default is *FALSE*. This must be set to *TRUE* for the view to use file or function data sources.

- **PERIOD**

  Determines the period for automatic updating, measured in milliseconds. A value of 0 means that the Plug-In does not periodically update. The default is 100.

- **SERVERPATH**

  This is a path to search for views, subdrawing, or text-based data source and data initialization scripts on the server. Only one path is allowed.

- **SRC**

  The URL of the view to display in the Plug-In instance. This is referred to as the top-level view.

- **STRETCH**

If *TRUE*, the drawport (rectangle in the browser screen) is created with TdpCreateStretch() ; otherwise *TdpCreate()* is used.

- **TARGET**

  Specifies a target for data returned by a DATASINK . Normally, this attribute is not set, which results in the *DATASINK* output being sent directly to the Plug-In for processing as a command stream. However, to observe the output of the *DATASINK* for testing and debugging purposes, you can set *TARGET* to *_current* or *_blank* or any other named frame or window. *_current* places the output in the current window, while *_blank* displays the output in a new window.

Appendix B: Troubleshooting / FAQ

This appendix contains remedies for common problems using the Plug-In.

- My View will not Load in the Browser Window.
- I changed my view on the server, reloaded, and it did not change. I then cleared Netscape's memory and disk caches, I still don't see the change.
- When I Run My View With a Data Sink and a Debug Trace File, the File Contains Errors from the Data Sink, but Everything Works Fine.

My View will not Load in the Browser Window.

If you cannot load a view into a browser window, consider the following possibilities:

- Did you choose a view file with a *.dvv* or *.dvweb* extension? If not, change the extension and try again.
- Are both DataViews and the DataViews Plug-In properly licensed? If you have not authorized each product, the Plug-In will not work.
- Did you load a binary view file from a different operating system into the Plug-In? If so, you only see the copyright screen.

I changed my view on the server, reloaded, and it did not change. I then cleared Netscape's memory and disk caches, I still don't see the change.

Most Web servers cache frequently accessed files independently of Netscape. You are probably seeing a version of your view that was cached at the server. There are two ways for the Web developer to avoid server caching problems:

- Create Web pages locally.

  This way, choosing "reload" always gives you the most current version of the file. This is not practical when developing Plug-In views with CGI data sources or data sinks since you need a server to test them.

- Run your own local Web server.

  This gives you complete control of the client-server testing environment. For example, you could guarantee seeing changes to views by turning off server file caching and Netscape file caching.

When I Run My View With a Data Sink and a Debug Trace File, the File Contains Errors from the Data Sink, but Everything Works Fine.

There might be a case where your data sink does not return any output to the Plug-In. If you send an empty header back to the Plug-In, this type of error does not occur.

Appendix C: Setup for Common Web Servers

For complete information about setting up your web server to serve DataViews views, Consult your web server documentation.. This Appendix contains summaries of how to set up some of the most common web servers.

Apache and NCSA Web Servers

CERN Web Server

Netscape Web Server

Microsoft Web Server

Apache and NCSA Web Servers

CERN Web Server   Netscape Web Server   Microsoft Web Server

The Apache server is meant to be a drop-in replacement for the NCSA server, so the following steps are the same for both servers.

- Make sure you have the necessary permissions to edit the Apache or NCSA server configuration files. In some cases you may need to be the user who installed the Apache or NCSA server on your system.
- Change directories to *<server_home>/httpd/conf* .
- Edit the file *mime.types*.
- Add the following entry somewhere in this file:
  ```
  text/x-dvweb dvv dvweb v
  ```
- Save the file.
- Restart the server. One way to restart the server is by executing the following command:
  ```
  kill -HUP pid
  ```
  where pid represents the process id of the Apache web server.

You have now registered the appropriate MIME types for viewing DataViews views on the Apache or NCSA web servers.

CERN Web Server

Apache and NCSA Web Servers   Netscape Web Server   Microsoft Web Server

To register the DataViews MIME type on your CERN server, use the following procedure:

- Make sure you are planning to make this change as a user with the necessary permissions to edit the CERN server configuration file. In some cases you may need to be the user who installed the CERN server on your system.
- The CERN server typically works with a single configuration file, which is */etc/httpd.conf* by default, although you may elect to have your configuration file somewhere other than */etc*. In the steps below, we assume your configuration file is under */etc*. If it is not, simply replace */etc* with the path to your appropriate configuration file.
- Edit the following file:

  ```
  /etc/httpd.conf
  ```
- Add the following lines to the end of the config file:

  ```
        AddType .dvv text/x-dvweb
        AddType .dvweb text/x-dvweb
        AddType .v text/x-dvweb
  ```
- Save the file.
- Restart the server. This can be done by either executing

  ```
  kill -HUP p_id
  ```
  where *p_id* represents the process id of the running CERN server, or you can restart the server by executing the following command:

  ```
  httpd -r /etc/httpd.conf -restart
  ```

Remember, the *httpd.conf* must be preceeded by the correct absolute pathname on your system to your configuation file.

You have now registered the appropriate MIME types for viewing DataView views on the CERN web server.

Netscape Web Server

Apache and NCSA Web Servers   CERN Web Server   Microsoft Web Server

To register the DataViews MIME type on your Netscape server, use the following procedure:

- Make sure you are planning to make this change as a user with the necessary permissions to edit the Netscape server configuration files. In some cases you may need to be the user who installed the Netscape server on your system.
- Change directories to your *<netscape_home>/httpd-80/config* directory.
- Edit the following file:

  `mime.types`

Add the following entry somewhere in this file:

```
type=text/x-dvweb exts=dvv dvweb v
```

- Save the file.
- Restart the server. One way to restart the server is by running the restart program, which is found under *<netscape_home>/httpd-80/*.

You have now registered the appropriate MIME types for viewing DataViews views on the Netscape web server.

Microsoft Web Server

Apache and NCSA Web Servers   CERN Web Server   Netscape Web Server

To register the DataViews MIME type on your Microsoft server, use the following procedure:

- Start *Regedit.exe* and open
  `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Inetinfo\Parameters\MimeMap`
- Click on "MimeMap."
- Click on "Edit value."
- Enter the following value for the MIME mapping:
  `text/x-dvweb,dvv,,0`

  Repeat the "Edit Value" process to add:

  ```
  text/x-dvweb,dvweb,,0
  text/x-dvweb,v,,0
  ```
- Restart the server.

You have now registered the appropriate MIME types for viewing DataViews views on the Microsoft web server.