DataPool User Guide

Version 4.0

Disclaimer

Vizrt provides this publication "as is" without warranty of any kind, either expressed or implied. This publication may contain technical inaccuracies or typographical errors. While every precaution has been taken in the preparation of this document to ensure that it contains accurate and up-to-date information, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained in this document. Vizrt's policy is one of continual development, so the content of this document is periodically subject to be modified without notice. These changes will be incorporated in new editions of the publication. Vizrt may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time. Vizrt may have patents or pending patent applications covering subject matters in this document. The furnishing of this document does not give you any license to these patents.

Technical Support

For technical support and the latest news of upgrades, documentation, and related products, visit the Vizrt web site at www.vizrt.com.

Created on

2019/08/14

Contents

1 Introduction

This document is the User Guide for DataPool. DataPool is a mechanism for managing data in Viz, which is implemented through a group of container function plugins and a group of scene plugins added to the Viz installation.

1.1 Related Documents

- Viz Artist User Guide: Contains information on how to create graphics scenes in Viz Artist.
- Viz Engine Administrator Guide: Contains information on how to install the Viz Engine software and supported hardware.

1.2 Feedback And Suggestions

We encourage suggestions and feedback about our products and documentation. To give feedback and/or suggestions, please contact your local Vizrt customer support team at www.vizrt.com.

2 DataPool

DataPool is a mechanism for managing data in Viz. The mechanism is implemented through a group of container function plugins and a group of scene plugins added to the Viz installation.

DataPool implementation allows Viz to keep information in named cells called DataFields. The data is managed at a separated, virtual, work space, linked to the graphics thru a set of plugins that make the connection between the DataFields and the graphic objects.
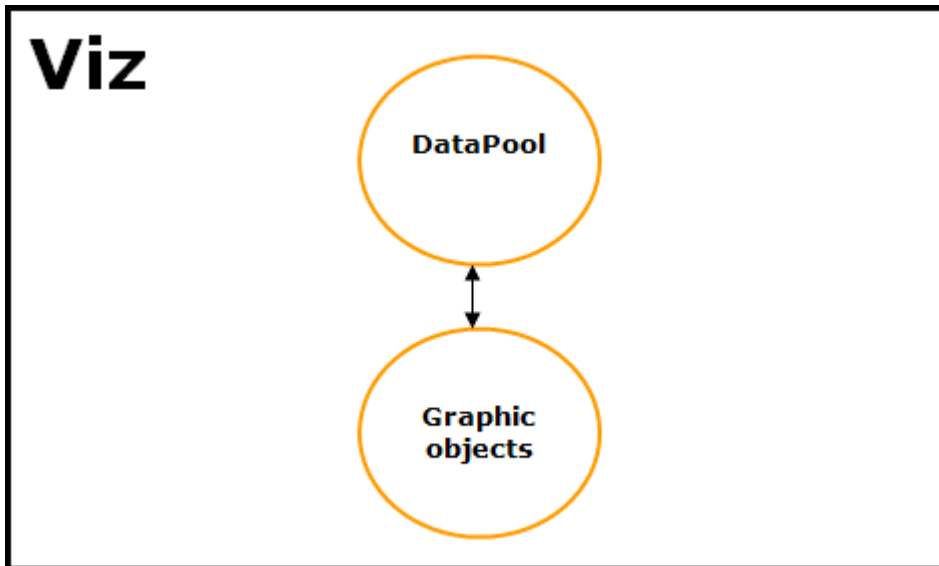
Some of the DataPool plugins have no effect over the graphics and are used only for managing and manipulating data. Some DataPool plugins are used for assigning data to DataFields and some plugins control Viz behavior upon changes in DataField values, affecting the graphic objects.

The native communication protocol to Viz is command oriented. Any change of the graphic objects, requires a separate command that is sent to the renderer. Each command includes the path to the container and the operation performed on the container.

2.1 Example

In a scene with a cube object, the cube can be scaled, moved, its bevel parameter changed, etc.

This is done by sending a command to Viz that specifies the name of the cube, its location in the Viz scene tree, the parameter to be changed and the new value for the parameter. If we add a font object to the same scene, we would be able to modify different parameters of the object, like setting its font, contents, kerning and so on.

If we want to implement a histogram of one single bar formed by the cube and the text object, setting the bar to the value `10`, we need to send Viz the following commands:

- Set the font object to display the string `10` (maybe add a suffix or prefix like a dollar sign: `$10`).
- Calculate the size of the cube and send the correct scaling value command Viz.

To change the value of the bar, two commands were sent, containing a different format for displaying the same graphic information: setting the bar to the value `10`.

Another issue is that each external command sent to Viz contains the address of the container to which it is addressed. If a software program needs to send the two commands to Viz, it has to know the names of the containers and their location in the scene hierarchy. If an external application is written controlling a scene, the scene hierarchy and container names cannot be changed in the scene.

All these issues mentioned previously create difficult maintenance problems. DataPool is designed to solve these problems. It extends the external command mechanism of Viz, allowing the user, or the controlling software, to send data rather than container specific operations. The data is addressed to named fields (DataFields). The scene designer uses special purpose plugins, used for processing and handling the incoming data.

If we modify the previous example to use the DataPool mechanism, a DataText plugin is added to the text object and a DataScale plugin is added to the cube object. Both plugins are registered to receive data from the same DataField named *Bar Value*. As soon as *Bar Value* changes both plugins react, the first changing the string of the text object and the second scaling the cube. The data is sent to the scene through a scene plugin called DataPool. This plugin is just an interface to enter the data. Once this plugin is defined in the scene, any external software can address it without the need of knowing the scene hierarchy or the container names.

3 Installation

DataPool is installed into the plugin directory under the last Viz installation. The plugins are installed according to the version found on the machine. The plugins are divided into four Plugin Groups.

This section contains information on the following topics:

- Plugin Groups
- Installing DataPool
- Advanced Silent Installation

3.1 Plugin Groups

The plugins are divided into four groups:

- DataPool
- Interactive
- Project Specific
- Script

For most users, installing the DataPool group is sufficient. For advanced applications install other plugin groups as well.

The installation includes the `DataPoolLib.dll` and an empty config.dp file which are installed in the Viz directory. When installing the DataScript plugin, `js32.dll` is also installed to the Viz directory.

3.1.1 DataPool

Main plugin group (default). Includes most of the plugins:

Data3DObject	DataKey	DataPosition
DataAction	DataKeyFrame	DataReader
DataActionTable	DataKeyFrame2	DataRotation
DataAlpha	DataKeyTime	DataScale
DataAnim	DataLUImage	DataSelector
DataArray	DataMaterial	DataSHM (Viz Engine 3.X only)
DataArrow	DataMaterialGradiant	DataSHMTracker (Viz Engine 3.X only)
DataClock	DataMaterialIndex	DataStructure

DataCondition	DataMaterialTable	DataSwitch
DataCountDown	DataMath	DataSystem
DataCounter	DataMinMax	DataTable
DataDevice (not installed for Viz 3.X)	DataMultiParam	DataText
DataDirector	DataNumber	DataTextKerning
DataFeedback	DataObject	DataTexture
DataGeom	DataObjectTracker	DataTime
DataGraph	DataParameter	DataTimer
DataGraphPoint	DataParamTracker	DataViz3Script (Viz Engine 3.X only)
DataImage	DataPool	

3.1.2 Interactive

All plugins that are based on mouse/keyboard events:

- DataClick
- DataFeedBack
- DataHyperLink
- DataInteractive
- DataKeyText
- DataManipulate
- DataMouseAction
- DataMousePosition
- DataMouseSensor

3.1.3 Project Specific

Plugins written for specific projects but can be used in other applications.

- DataCamera
- DataDispatcher
- DataDrawMask
- DataScreen

3.1.4 Script

Contains the DataScript plugin and the script library.

· DataScript (Viz Engine 2.X only)

3.2 Installing DataPool

3.2.1 First-time Installation

1. Run the installation file and follow the instructions. When installing DataPool for the first time three installation types are available:
   · **Typical:** Installs only the DataPool plugin group.
   · **Custom:** Installs the plugin groups that the user selects.
   · **Full:** Installs all the DataPool plugins (all the groups).
2. Select the installation type and continue. An information window displays the path of the installation and selected plugin groups.

   ⚠ **Note:** The plugins are installed to the plugin directory under the latest Viz installation.

   ⚠ **Note:** This User Guide is installed with the DataPool group.

3.2.2 To Upgrade an Existing Installation

1. When upgrading the installation three options are displayed:
   · **Modify:** Changes the installed groups on the machine.
   · **Repair:** Reinstalls the installed groups (*.dp* files are not overwritten).
   · **Remove:** Uninstalls DataPool.
2. Select one of the options and click the next button.

3.3 Advanced Silent Installation

The advanced silent installation section describes how to create a custom silent installation, should the default silent installation not fit the user requirements. One manual run of the installer is required to record user actions. After the sequence of user actions is recorded, the installer can run in silent mode, repeating the recorded user actions.

⚠ **Note:** The procedure described here, using an application shortcut, can be executed from a command shell running the installation file with the described flags.

3.3.1   Recording user actions

1. Create a shortcut for the installer .exe file.
2. Open the shortcut properties window.
3. Add the following flags to the **Target field** in the properties window:

```
/r /f1"My_File_Name.iss"
```

where `My_File_Name.iss` is a qualified path and file name with the suffix .iss. The path to the file should be a full path or a relative path and it should be double quoted. There should be a space between the `/r` flag and the `/f1` flag, but there is no space between `/f1` flag and the specified file name. For example:

```
/r /f1"C:\Temp\VCMC.iss" or "/r /f1".\VCMC.iss"
```

4. Click the **OK** button and run the shortcut by double clicking it.
5. Install Viz DataPool manually as described in the User Installation section above (First Time Installation section). Make the selections to comply with the required silent installation flow.
6. Check that the specified .iss file was created.

3.3.2   Running the installer in silent mode

1. Create a new shortcut for the installer .exe file.
2. Open the shortcut properties window.
3. Add the following flags to the **Target** field in the properties window:

```
/s /f1"My_File_Name.iss"
```

where My_File_Name.iss is the qualified path and file name to the .iss file that was recorded in the previous section.
The path to the file should be a full path or a relative path and it should be double quoted. There should be a space between the `/r` flag and the `/f1` flag, but there is no space between `/f1` flag and the specified file name. For example:

```
/s /f1"C:\Temp\VCMC.iss" or "/s /f1".\VCMC.iss"
```

4. Click the **OK** button and run the shortcut by double clicking it.
5. The installation runs without any dialogs.
6. Check that the required files were installed.
7. Continue to install all clients using the silent mode option.

> ⚠ **Note:** When running the installer in silent mode when DataPool is already installed, the installed components on the machine are reinstalled.

4 Architecture

This section describes the architecture of DataPool and contains information on the following topics:

4.1 Overview
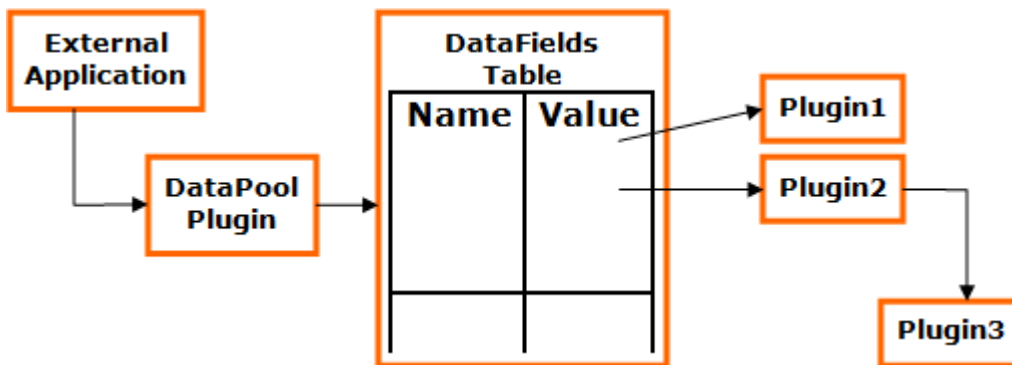
DataPool implements a table of DataFields. Each DataField has a unique name by which it is accessed, a value (data) and a list of data changes handlers. The handlers are implemented as plugins in Viz, where each plugin affects specific parameters of a container in Viz. The plugin instances are registered to DataFields when attached to a container in the scene. The plugin modifies the container parameters when the value of the DataField changes. It doesn't do anything unless it is called.

The external software sends the data to the DataPool through the DataPool plugin. In the sent message, the application specifies which data should be assigned to which DataField. The DataPool plugin accesses the DataFields table using the name of the field to be accessed, gets the relevant field and assigns the data to it.

This assignment invokes a series of triggers to all the plugins (handlers) registered to the field.

4.1.1 DataFields

As stated before, the DataField is a cell formed by three elements:

- Name (or address).
- Data.
- A list of handlers. The name is a zero terminated ASCII string.

The data is a vector of strings. The DataField can host one string of data, or a series of strings. The user can set the size (the number of items) of the DataField. Each one of the DataField items can be accessed separately or they can be accessed as a group, altogether.

The process of creation of the DataField is as follows: When a DataPool plugin is specified to register to a certain DataField the plugin looks for the specific DataField in the DataFields table. If the field doesn't exist the plugin creates it and registers itself to the DataField. In order to send data to a certain DataField, at least one plugin has to be registered to that DataField.

4.2 DataPool Hierarchy

4.2.1 DataNodes and DataFields

DataPool is a hierarchical data structure composed of DataFields and nodes. DataFields are the lowest level of the hierarchy. The DataFields don't have children. Nodes are groups of DataFields. The DataPool hierarchy has a root node. Each of the DataFields in the hierarchy has a parent node, either the root node or any other sub-level node.

4.2.2 Example 1

The following example shows a hierarchy with a root, two nodes (NA and NB) and six DataFields. As can be seen each of the DataFields reside below a node. Setting the value of the DataField FA under the node NA follows the hierarchical structure to the required DataField:

```
NA/FA=<some information>;
```

In the case of field FC under NB:

```
NB/FC=<some other information>;
```
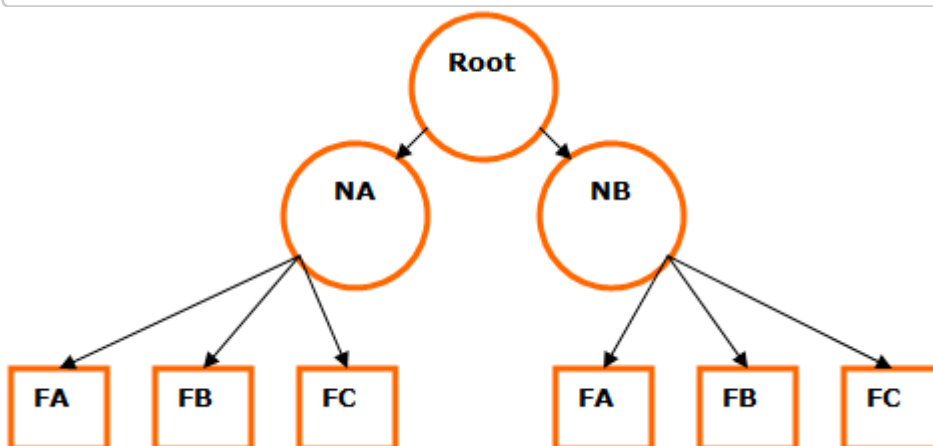

To set the data of all DataFields under the node NB, the following commands are sent:

```
NB/FA=<info for A>;
NB/FB=<info for B>;
NB/FC=<info for C>;
```

Note that there is a certain amount of redundancy because NB appears three times. To avoid using the node name every time, the following alternative format is used:

```
NB={
        FA=<info for A>;
        FB=<info for B>;
        FC=<info for C>;
};
```

The brackets determine the scope of the contained statements. In the above command the statement "FA=<info for A>;" is in the scope of object NB.

4.2.3 Example 2

Consider the following example:
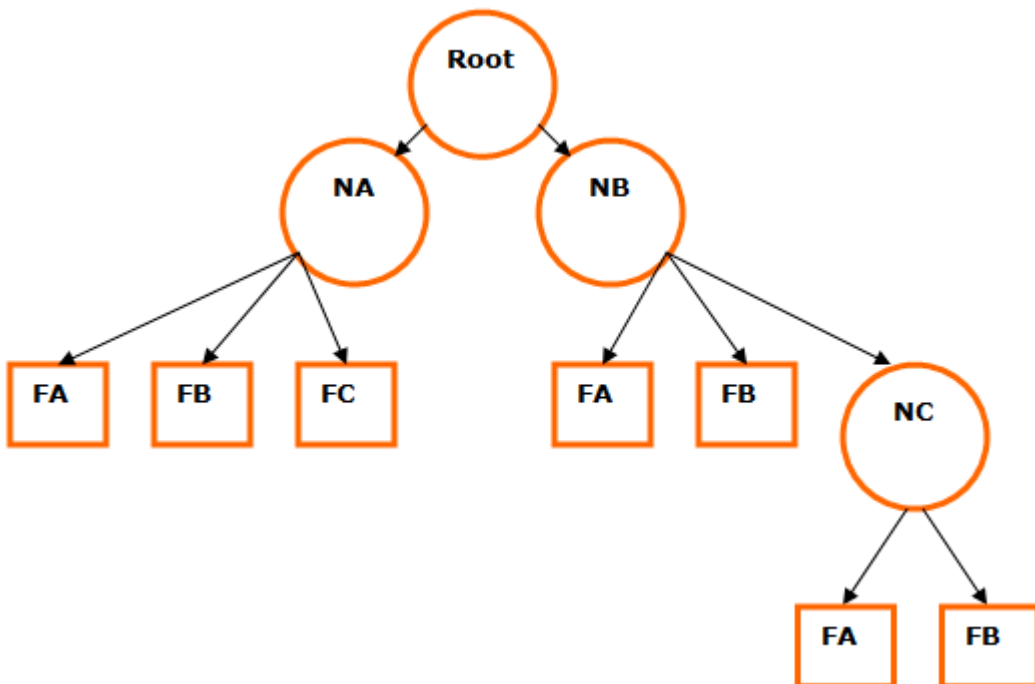
In this example NB has a child node called NC. To set data to the field FB under NC the following command is sent:

```
NB/NC/FB = <info for B>;
```

In order to populate all the fields in NC:

```
NB/NC= {
        FA = <info for A>;
        FB = <info for B>;
};
```

If we want to populate the whole hierarchy with data in one command:

```
NA = {
        FA =<info for A>;
        FB=<info for B>;
        FC=<info for C>;
};
NB = {
        FA =<info for A>;
        FB=<info for B>;
        NC = {
                FA=<info for A>;
                FB=<info for B>;
        };
};
```

4.3 Data Objects

Data objects use the structure of DataNodes and DataFields to describe properties of objects used for displaying information in Viz scenes.

An **example** of such an object is a stock. A stock object can be represented in the following way:
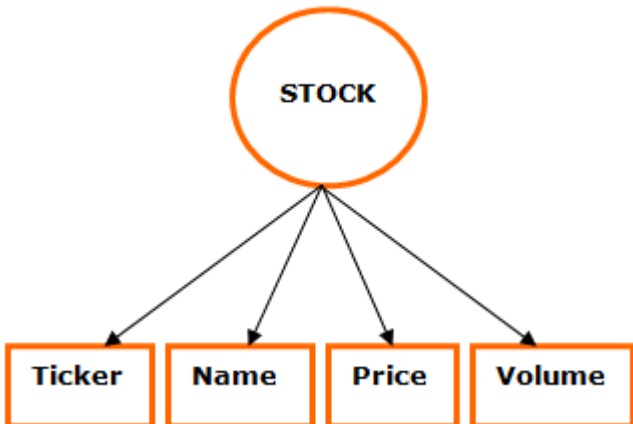
The DataPool representation of this stock object would be:

```
Stock={
        string Ticker;
        string Name;
        string Price;
        string Volume;
};
```

The meaning of this definition is: A Stock object is defined by four fields: Ticker, Name, Price and Volume.

The type string preceding each property of the stock means that incoming data is a text string and should be handled as such by the DataPool plugins.

To send information to an object called MyStock, defined as a Stock object, the following format is used:

```
MyStock={
        Ticker=IBM;
        Name=International Business Machines;
        Price=85.26;
        Volume= 5,188,500;
};
```

This information is sent to the DataPool plugin and used by the different DataPool plugins in the scene. Other examples for objects are cities weather conditions, election results, etc.

DataPool object definitions are saved in configuration files with the special suffix .dp, saved under Viz folder. DataPool loads all the objects defined in all of the .dp files, and makes the object types defined available to DataPool plugins such as DataObject or DataArray.

The default configuration file is called config.dp and it is installed empty during DataPool installation. When upgrading DataPool, the config.dp file is never overwritten. Multiple configuration files can be used to define various objects. All files must be saved with the .dp suffix.

4.4 Arrays Of Data Objects

When dealing with multiple instances of the object type, an array of object types can be defined. Defining an array of objects simplifies data management when sending the information to the different objects.

Sometimes there is a need to maintain lists of objects. An example could be the display of a histogram of stocks. Each Viz object is attributed a DataPool object. But in this case, it would be very unwise to give a name to each of the objects because the number of the objects may be unknown. DataPool solves this problem by providing with a means of array of objects. Like in the case of the single object, the array has an object type. When defined the array builds a set of objects as its children. This scheme is shown in the following figure:
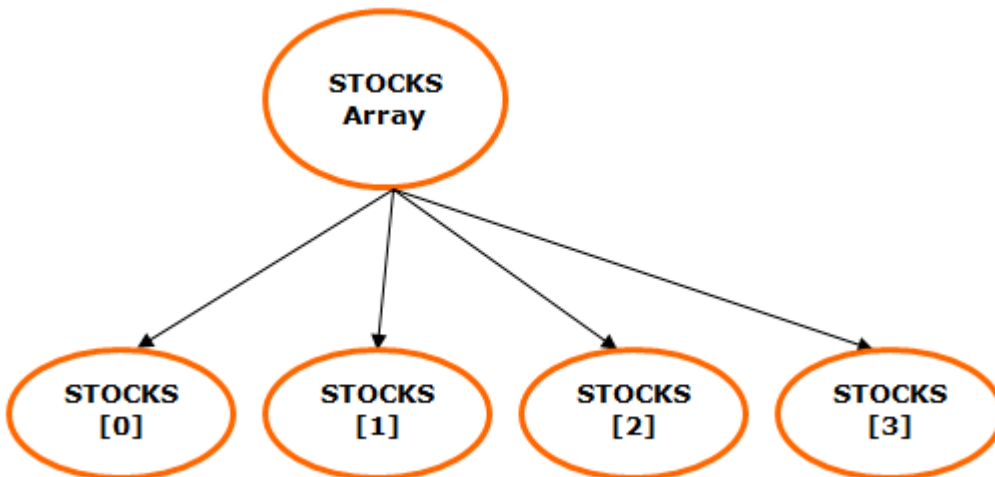
In the figure, an array of stocks of size four is shown. As can be seen each of its children are named Stocks[0], Stocks[1], and so on. To enter data to the array element number 2:

```
Stocks[2]/Price=85.3;
```

If we want to enter information to the whole array:

```
Stocks[0-3]={
        {Ticker=IBM;Price=85.26;},
        {Ticker=SGI;Price=1.09;},
        {Ticker=AOL;Price=14.71;},
        {Ticker=AA;Price=23.85;},
};
```

> ⚠ **Note:** There are two sets of brackets in the above statement: One for the array itself and a second for the objects themselves. Also note that because this is an array, the objects are separated by commas rather than by semicolons.

4.5 Tables Of Data Objects

In the DataArray plugin, we can organize the objects in an array structure and refer to the whole array or to single objects within the array using the array name and an index.

Sometimes it is useful to organize the objects and keep the object's names. For example, rather than keeping a list of stocks we could keep a table in which IBM's stock is stored under the name IBM, SGI's stock under the name SGI and so on. As in the array, the table has an object type, but unlike the array the table doesn't have a size. Every time an object is inserted in the table the table grows. The DataTable plugin stores the data sent to it and it is able to recall this data when requested. That data can later be used with the copy functionality of DataPool, to effect the graphic objects thru the DataFields.
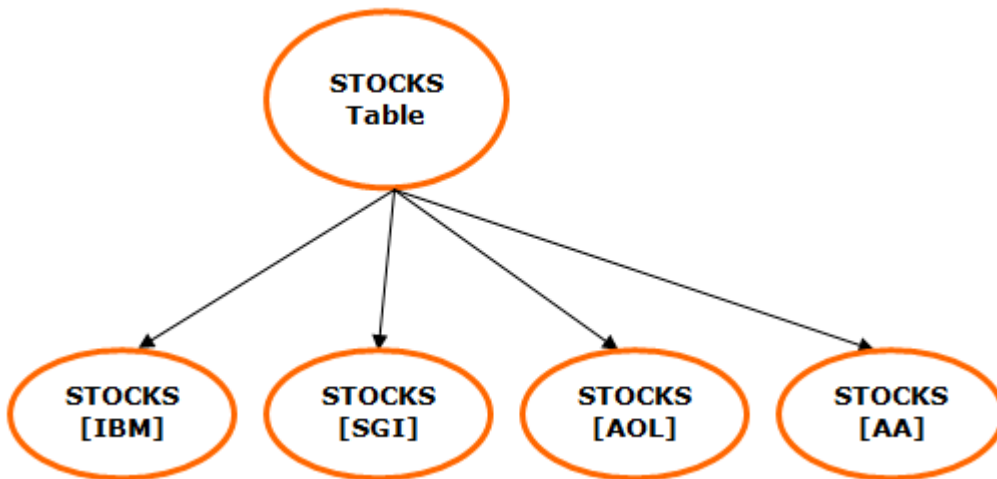
A typical table of stocks could look as follows:

In the above figure a table of stocks named "Stocks" can be seen.

The children objects are called "Stocks[IBM]", "Stocks[SGI]", etc…

To send information to the stock IBM:

```
Stocks[IBM]/Price = 85.26;
```

or

```
Stocks[IBM]={
        {
                Volume=5,188,500;
                Price=85.26
        ;}
};
```

To enter information to a number of stocks:

```
Stocks[IBM, AOL]={
        {
                Ticker=IBM;
                Price=85.26;
        },
        {
                Ticker=AOL;
                Price=14.71;
        }
};
```

4.6 DataPool Variables Scope

When referring to DataPool variables, the common and wide spread meaning is addressing global variables by their name and addressing structured variables by specifying the full path to the variable (i.e. *<structure name>/../<variable name>*).

In some DataPool plugins (where applicable), one of the parameters defines the DataPool variable scope. The options are Local and Global:

- **Local Variables:** When referring to Local DataPool Variables, the meaning is variables within a DataPool object/data structure. When the scope is set to Local, the DataPool variable is addressed by its name only, even when it is a part of a DataPool structure.
- **Global Variables:** When referring to Global DataPool Variables, the meaning is variables that are not defined as part of a DataPool structure.

> ⚠ **Note:** This parameter only has an effect on DataPool expressions and variables that reside under the DataObject container, i.e. child containers within the object's hierarchy.

4.6.1 Example

The following definitions are set in the `config.dp` file:

```
string MyText;
MyObject{
        string MyText;
        string TextLength;
};
```

When addressing the variable MyText:

- MyText="This Text"; → the Global variable MyText is set to *This Text*.
- MyObject/MyText="This Text"; → The variable MyText in the MyObject structure is set to *This Text*.
  When the scope parameter is set to Local and setting the following assignment in one of the structure's children MyText="This Text"; then the effect is like defining the full path: MyObject/MyText="This Text";.

4.7 DataPool Configuration Files

This page contains information on the following topics:

- DP Configuration Files
  - Example Configuration File
- DataPool.ini File
- Conversion Table

- [File syntax](#)

> ⬧ **Warning**: If using Viz Engine 4 or newer: When editing DataPool configuration files, always modify the files located in *%ProgramData%/vizrt/VizEngine*! The files in *\Program Files* are just templates which are copied to *%ProgramData%* at installation time. The Engine reads the files from *%ProgramData%*.

4.7.1 DP Configuration Files

DataPool data fields and objects are used across scenes and engine machines to support Viz scenes and data distribution. The DataPool configuration files, used to define DataPool structures and DataPool variables, use the suffix .dp. All files with .dp suffix are loaded during Viz load and the data fields and structures are defined in the DataPool memory. The configuration files can be copied to all relevant Viz machines or saved in a common folder and defined in the *DataPool.ini* file.

When installing DataPool, the file *datapool.dp* is installed to the Viz Program Files folder and copied to *%ProgramData%/vizrt/VizEngine*. This file is an example and the users can modify the file to define data fields. Additional .dp files can be used to organize the DataPool variables and data structures.

Example Configuration File

```
Example_STOCK = {
        string Example_ID;
        string Example_Name;
        string Example_symbol;
        string Example_Open;
        string Example_Close;
        string Example_High;
        string Example_Low;
        string Example_Volume;
        string Example_Price;
        string Example_TimeRange;
};
Example_RESULTS = {
        string Example_NAME;
        string Example_SCORE;
        string Example_COLOR;
};
```

The file shows two DataPool structures.

4.7.2 DataPool.ini File

The *DataPool.ini* file is installed to the Viz Program Files folder and copied to *%ProgramData%/vizrt/VizEngine*. It contains the following parameters:

- **ConfigurationFolder:** Defines the location of the `.dp` files to be loaded. If omitted, the default path is set to `.\` (i.e. Viz folder).
- **MultiDatapool:** Defines if a separate Viz DataPool segment is assigned to each scene loaded to Viz.

4.7.3 Conversion Table

The conversion tables file is a comma separated file installed in the Viz Program Files folder and copied to *%ProgramData%/vizrt/VizEngine*. All the conversion tables are defined in this file name, using a fixed format. The tables are used to convert input data values to another value defined in the table. The first column in the table is the input value and the second column is the converted value (output).

> ⚠ **Note:** The file name is hard coded: *DP_ConvTable.csv*.

File syntax

The file contains a few reserved tokens used for defining conversion tables and other parameters:

- **__VERSION__:** This entry is used for future file support and version compatibility issues.
- **__DELIMITER__:** This entry is used to define a different file delimiter (other than comma). If omitted comma is used.
- **__TABLE__:** This entry defines the beginning of a conversion table. The table end is defined by another table entry or end of file.
  Table example:

```
__TABLE__,colors,
red,255 0 0
green,0 255 0
blue,0 0 255
yellow,255 255 0
cyan,0 255 255
purple,255 0 255
__DEFAULT__, 0 0 0
,
```

  In the example, a table called colors is defined. When the conversion table is used in a plugin, an input string *red* is converted to the output string `255 0 0` and sent back to the plugin for processing.

- **__DEFAULT__:** This token is used in any of the conversion tables as a default conversion value (any value not found in the table is converted to the default value).

4.8 External Interface

The external interface to DataPool is based on Viz messages sent to the DataPool plugin. Each message contains a series of data assignments to different DataFields.

The structure of the message is:

```
command1 command2 command3 ...
```

The structure of each command is as follows:

```
<Name>[items range] = datum1, datum2, datum3, ...;
```

Where `name` is the name of the DataField to assign data to and `items range` is a description of items to which the data is addressed.

> ⚠ **Note**: The semicolon is part of the command and should be added at all times. It is used to define the end of the command.

The item ranges are a comma-separated list of indexes, or a from-to index.

4.8.1   Examples

1. Send data to a field named **VALUE**:

```
VALUE=30;
```

2. Send data to the first five values of a field called **VALUES**:

```
VALUES[0-4] = 0, 1, 2, 3, 4;
```

The above is equivalent to sending:

```
VALUES[0]=0; VALUES[1]=1; VALUES[2]=2; VALUES[3]=3; VALUES[4]=4;
```

> ⚠ **Note:** The indexes of the items are zero-based (start from 0).

3. Send data to the items 1-3, 55-57 and 101:

```
VALUES[1-3,55-57, 101] = 0, 1, 2, 3, 4, 5, 6;
```

This is equivalent to sending:

```
VALUES[1]=0; VALUES[2]=1; VALUES[3]=2; VALUES[55]=3; VALUES[56]=4;
VALUES[57]=5; VALUES[101]=6;
```

5 Plugins

The DataPool plugins in Viz are responsible for handling incoming data and for affecting the objects in the scene according to the DataFields values.

5.1 Overview

The DataPool plugins are found in Viz in the Function plugins pool, under a folder called **Data**. There are DataPool Scene Plugins and DataPool Container Plugins. The scene plugins manage incoming data from different sources, inserting it to DataFields and maintain the DataFields Table. For example, the DataPool plugin receives information from external sources and inserts it to the different DataFields. The DataMouseSensor senses mouse position and events and stores them in predefined DataFields.

The container DataPool plugins trigger graphic changes in the container they are attached to, when the DataField registered to them changes. Most of the container plugins have an argument called **Field Name**. The field name value is the name of the DataField to which the plugin is registered to. The Field Name can take the following forms:

```
<name>
<name>[number of items]
```

If the field name is **VALUE** then the DataField is defined as a single datum field. However, if the field name is VALUE [100] then the name of the DataField is **VALUE** and it has 100 data items (values). The field name, scene hierarchy and the way the information is sent to the DataPool plugin determine the container plugin behavior.

5.2 Example

This is a hierarchy of a graph scene:
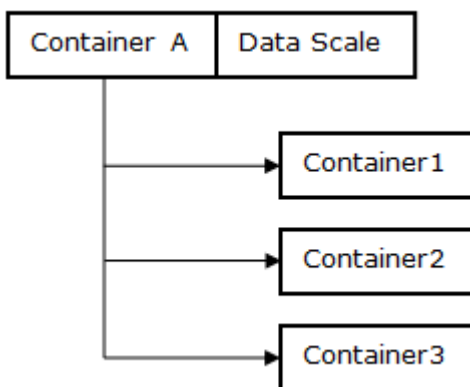
The above figure shows a container A with three children 1, 2, and 3. A DataScale plugin is attached to A. The DataScale plugin controls the scaling of the container according to the data it receives.

If the *Field Name* in the DataScale plugin is defined to be `VALUES[3]` (a vector of 3 items named VALUES), DataScale acts in two different ways:

1. If the data is entered without specifying an item index, DataScale affects container A.

   > ⓘ **Example:** If the data is entered as VALUES=0.5 0.5 0.5; then the container A is scaled by `0.5` in each axis.

2. If the data specifies an item index, DataScale works on the respective child container of A.

   > ⓘ **Example:** If the data is entered as `VALUES[2] = 0.1 0.1 0.1;`, then container3 is scaled by `0.1` in each axis (Note that the index value is zero based).

5.3 Expressions

An expression is a text parameter combining constant strings and references to DataPool variable's (DataField's) value. The reference syntax is `$(DataField)`. When using the expression, the string `$(DataField)` is replaced by the value of the DataField specified in the parentheses. For example, suppose we need to run different animation directors in Viz and we want to send the name of the director as a parameter. The command in Viz to run a director is:

```
0 RENDERER*STAGE*DIRECTOR*<name> START
```

So we need to send the entire command for each director that we want to run. By using expressions, only the name of the director to be run is sent to Viz. We define a DataField called DIR that contains the name of the director to run.

> ⚠ **Note:** Predefined DataFields are defined in the config.dp file, residing in the Viz folder.

Now we can use the following syntax for running the animations:

```
0 RENDERER*STAGE*DIRECTOR*$(DIR) START
```

$(DIR) is replaced by the incoming data, so if the data was DIR=mydirector; the whole expression is interpreted to be:

```
0 RENDERER*STAGE*DIRECTOR*mydirector START
```

> ⚠ **Note:** The string enclosed between the parentheses is considered to be either a DataPool variable, a special DataPool command or a special DataPool variable.

A common DataPool variable is **$(INDEX)**. This variable is used to access the full index range of a DataField. For example, if we have two DataFields called DST[10] and SRC[10] and we want to copy all the values from vector SRC to vector DST, the expression should state `DST[$(INDEX)]=SRC[$(INDEX)]`. A useful DataPool special command is **$(REFRESH)**. The effect of having an expression like `$(REFRESH) $(VALUE)` is to trigger all the plugins registered to the DataField called VALUE

without having to receive external data. Each data plugin has its own set of special purpose variables. They are explained in the explanation of each plugin.

5.4   Special DataPool Variables

- **$(INDEX):** Returns the indexes of sub-containers of the selected container.
- **$(REFRESH) $(DataField_Name):** Triggers all plugins registered to DataField_Name as if the DataField DataField_Name was changed.
- **$(SCENE), $(THIS_SCENE):** Returns the name of the current scene.
- **$(CONTAINER):** Returns the container ID of the container that the DataPool Plugin is attached to.
- **$(CONTNAME):** Returns the name of the container hosting the DataPool plugin.
- **$(CHILD):** Returns the index of the selected child.
- **$(DATA):** Returns the string value of a text object.
- **$(PARENT):** Represents the parent DataPool object/structure of the current DataPool field. This variable enables the user to change the values of its "brother" data fields. This special variable is applicable only in plugins that have an Action parameter.

  Example: In the given structure

```
DEMO={
        string Field1
        string Field2
};
```

If a DataPool plugin, with an Action parameter, on the Field1 container refers to $(PARENT)/ Field2, it sets the value of Field2 in the same structure:

```
$(PARENT)/Field2="This is the Field1" $(Field1)
```

The result of this action is that *Field2* contains the quoted text and the content of *Field1* data field.

> ❗ **IMPORTANT!** When working with DataPool structures/objects, the $(PARENT) is the recommended way of addressing other data fields in the DataPool structure/object.

> ⚠ **Note:** Using the $(PARENT) variable is similar to using the Global/Local parameter in some of the DataPool plugins. Both make the distinction between a DataPool variable that is part of a data structure/object and a DataPool variable that is used not a part of a structure/object.

> ⚠ **Note:** The Field Name parameter (i.e. the DataPool variable that triggers the action should be a part of the structure. The $(PARENT) value is defined according to this data field.

- **$(PARENT_NAME):** This special variable is applicable only in plugins that have an Action parameter. The $(PARENT_NAME) returns the parent DataPool object/structure name, of the current DataPool field.

> ⚠ **Note:** The Field Name parameter (i.e. the DataPool variable that triggers the action should be a part of the structure. The $(PARENT_NAME) value is defined according to this data field.

See $(PARENT) description for additional details.

- **$(PARENT_FULL_NAME):** Returns the full hierarchy string of the parent DataPool object/ structure, of the current DataPool field. If the structure/object has only one level of variables in the hierarchy, then the $(PARENT_FULL_NAME) and the $(PARENT_NAME) returns the same value. If the structure/object has more than one level, the $(PARENT_FULL_NAME) returns the entire path of the structure from the requested level. The full name is returned in the format of: xx/yy/zz/. This special variable is applicable only in plugins that have an Action parameter.

> ⚠ **Note:** The Field Name parameter (i.e. the DataPool variable that triggers the action should be a part of the structure. The $(PARENT_FULL_NAME) value is defined according to this data field.

See $(PARENT) description for additional details.

- **$(FIELD_NAME):** Returns the name of the DataField when used in a child container.
- **$(FIELD_FULL_NAME):** Returns the full name of the DataField when used in a child container.
- **$(FIELD_DATA):** Returns the value of a DataField.
- **$(SCENE_FULL_NAME):** Returns the full path and scene name.

5.5 Common Parameters For All DataPool Plugins

- **Field Name:** Defines the name of the DataField from which the plugin receives the data.
- **Data Min and Data Max:** Specifies value mapping between incoming data and a Viz value. These parameters are always used in conjunction with two parameters representing Viz values (usually named Value Min and Value Max). The Data Min value is converted to a minimum Viz value and the Max Data is converted to a maximum Viz value. All incoming values are converted respectively to Viz values. When the Clamp parameter is set `On`, Values out of this range (smaller than Data Min or larger than Data Max) is cropped to the Min/Max values. When Clamp is `Off`, incoming values are divided by the range between Data Min and Data Max and the remainder value is used as the incoming value and interpolated to Viz values.

```
## Example
Data Min is 0
Data Max is 100
Viz Min is 0
Viz Max is 100
Sent value is 125
```

If Clamp is `On` the Viz value is set to `100` (i.e. all values above Data Max (`100`) are cropped to the Data Max Value and interpolated to Viz Values)
If Clamp is `Off`, Viz value is set to `25`. Incoming value is `125`, divided by `100` (the range between Data Min and Data Max), the remainder is `25` and it is interpolated to Viz value.

- **Prefix:** Specifies a prefix string that is added in front of the incoming data string.
- **Suffix:** Specifies a suffix string that is added at the end of the incoming data string.

- **Clamp:** Clamps the incoming data values to the data min/max range when set to `On`. All DataPool plugins that use the Data Min and Data Max parameters have a clamp button. Values out of this range (smaller than Data Min or larger than Data Max) are cropped to the Min/Max values when the Clamp parameter is set `On`.
- **Notify only on value change:** Triggers all DataPool plugins registered to that DataField when a DataField value is changed. This happens also if the DataField was set to the same values. Notifies the data plugins only if the field value was actually changed when enabled. The default behavior is `Off` (as it was in previous DP releases, before adding this feature).
- **Use Other Container:** Allows one container to affect another container, or allows another instance of the same plugin to be used on the same container. DataPool container plugins affect the containers they are attached to, and cannot affect other containers directly. A common example of using this parameter is changing multiple keyframes in an animation. Since only one plugin can be attached to the container, dummy containers with DataKeyFrame plugin, using this parameter can be used to affect the remaining keyframes (This parameter makes the DataKeyFrame2 plugin unnecessary. The plugin is installed for backwards compatibility only). When enabled, additional options are also enabled:
  - **PARENT:** The DataPool plugin affects the parent container, i.e. the container residing above the current container in the scene hierarchy.
  - **GrParent:** The DataPool plugin affects the grandparent container, i.e. the container residing two levels above the current container in the scene hierarchy.
  - **GrGrParent:** The DataPool plugin affects the great grandparent container, i.e. the container residing three levels above the current container in the scene hierarchy.
  - **REMOTE:** When remote is selected another parameter, Remote Container, is enabled. Drag the container to be effected to the container place holder.

> ⚠️ **Note:** When using the Parent, GrParent or GrGrParent options, the link to the other container is relative. When using the Remote option the link to the other container is absolute. Using the relative options is recommended when using object designs that are duplicated in the scene and controlling containers within the object hierarchy (like in Viz Curious Maps Client labels or Viz Weather objects).

> ⚠️ **Note:** The REMOTE option maintains compatibility with previous versions of DataPool using the Use Remote Container option.

- **Incremental Change:** Specifies if the incoming data is added to the current value of the data field (`On`) or replaces the current value of the data field (`Off`).
- **Use Conversion Table:** Enables a Conversion Table Name parameter when enabled. Enter the table name as it appears in the `DP_ConvTable.csv` file, without the proceedings leading and proceeding underscore characters.

6 Plugins Reference

This section contains details for each of the plugin types:

- Scene Plugins
- Container Plugins

6.1 Scene Plugins

The scene plugins are listed in alphabetical order:

- DataClock
- DataInteractive
- DataMaterialTable
- DataMouseSensor
- DataPool

6.1.1 DataClock

☐

DataClock plugin is a scene plugin that sends the current system time to predefined DataPool DataFields.
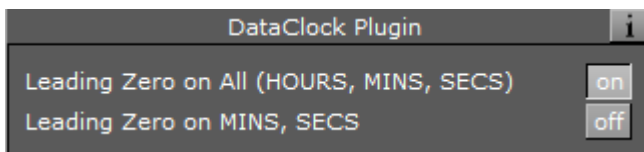
Unique Parameters

- **Leading Zero on All:** Adds a leading zero to all the single digit numbers (0-9) of hours, minutes and seconds.
- **Leading Zero on MINS, SECS:** Adds a leading zero to the single digit numbers (0-9) of minutes and seconds only.

Fields

The fields DataClock generates are:

- **DAY:** Current day (`1-31`).
- **MONTH:** Current month (`1-12`).
- **YEAR:** Current year (four digits).
- **WEEKDAY:** The day of the week (`1-7`).
- **DAY_NAME:** The textual name of the day.
- **MONTH_NAME:** The textual name of the month.
- **HOURS:** Hour of the day (`0-23`).
- **MINS:** Minute of the hour (`0-59`).

- **SECS:** Second of the minute.
- **AMPM:** AM or PM.
- **COUNTER:** A running counter of seconds. Its value is equal to 60*60*HOURS + 60*MINS + SECS.

6.1.2   DataInteractive

☐

DataInteractive plugin receives mouse and keyboard events and triggers actions accordingly. The actions are DataPool commands. This plugin is a scene plugin and it gets the mouse events from Viz (DataMouseSensor is not required).

Notes

- The interactive events are sent only when Viz is in On Air mode.
- When using DataInteractive in a scene and no other data is sent externally, adding the DataPool scene plugin is unnecessary.

· When using DataInteractive plugin, DataManipulate and DataClick cannot be used.
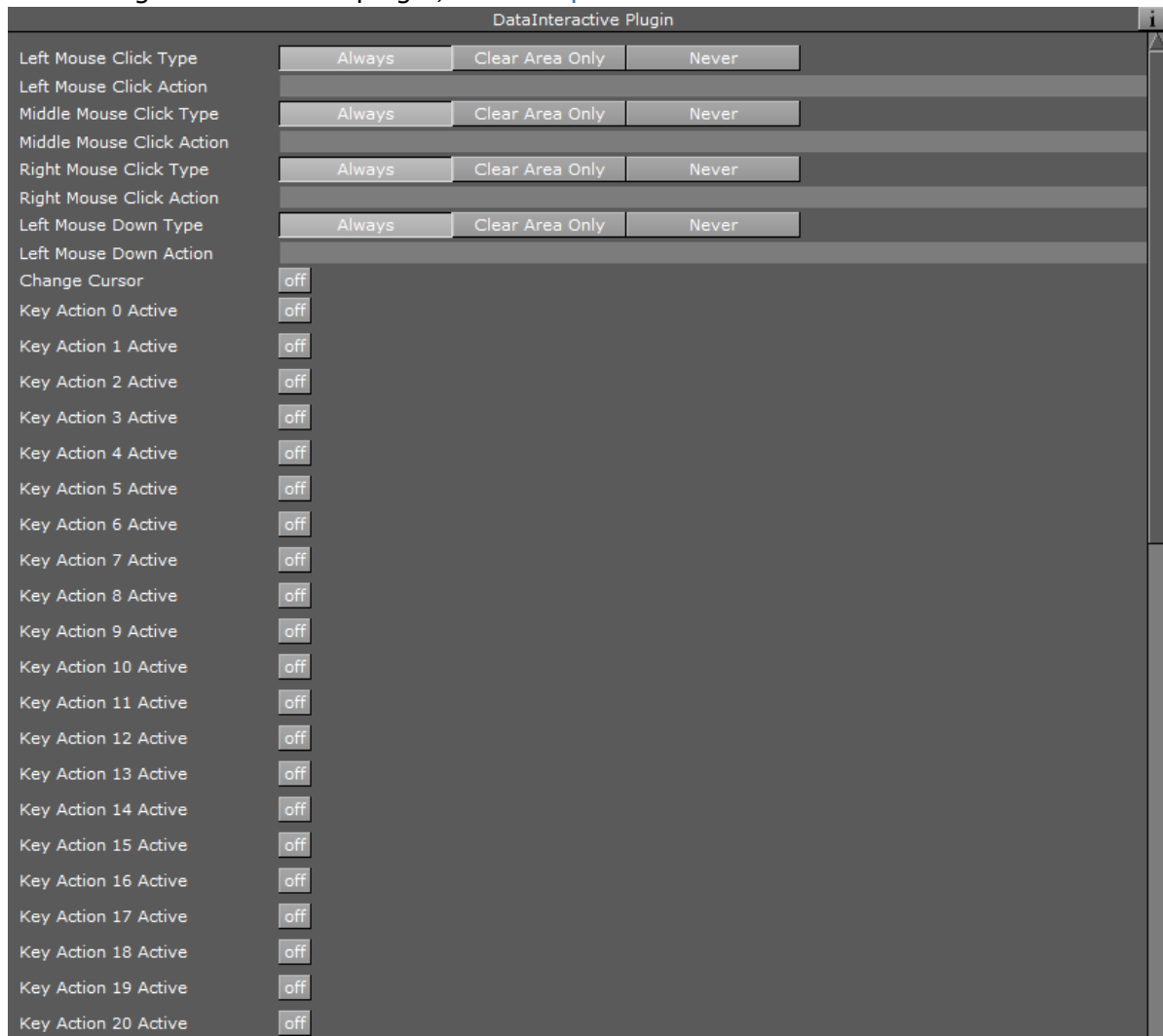
Unique Parameters

· **Click Type Left:** Executes the action defined in the Click Action Left Mouse parameter every time the left mouse button is clicked when set to `Always`. When set to `Clear Area Only`, the action is triggered if the mouse is clicked on the background (not on any container area). When set to `Never`, the action is disabled.

· **Click Action Left Mouse:** Defines a DataPool action that is performed when the mouse is clicked.

· **Click Type Middle:** Executes the action defined in the Click Action Middle Mouse parameter every time the left mouse button is clicked when set to `Always`. When set to `Clear Area Only`, the action is triggered if the mouse is clicked on the background (not on any container area). When set to `Never`, the action is disabled.

· **Click Action Middle Mouse:** Defines a DataPool action that is performed when the mouse is clicked.

- **Click Type Right:** Executes the action defined in the Click Action Right Mouse parameter every time the left mouse button is clicked when set to `Always`. When set to `Clear Area Only`, the action is triggered if the mouse is clicked on the background (not on any container area). When set to `Never`, the action is disabled.
- **Click Action Right Mouse:** Defines a DataPool action that is performed when the mouse is clicked.
- **Mouse Action 0 (to 9) Active:** Enables additional parameters when set to `On`. The action is disabled when set to `Off`.
    - **Alt For Action 0 (to 9):** Defines if the **Alt** button must be pressed in conjunction with the defined key in the parameter Key For Action 0 (to 9).
    - **Ctrl For Action 0 (to 9):** Defines if the **Ctrl** button must be pressed in conjunction with the defined key in the parameter Key For Action 0 (to 9).
    - **Shift For Action 0 (to 9):** Defines if the **Shift** button must be pressed in conjunction with the defined key in the parameter Key For Action 0 (to 9).

6.1.3 DataMaterialTable

☐

The DataMaterialTable plugin enables the user to create a table of materials and store this table to a file, enabling the same table to be used in different scenes. When a table is defined at the scene level (by loading a table from a file or defining material entries) it is accessible by the DataMaterialIndex plugin.

> ⚠ **Note:** The material table has a maximum of 256 entries. Index range is `0-255`. When saving/reading the material table file all the entries are saved/read.
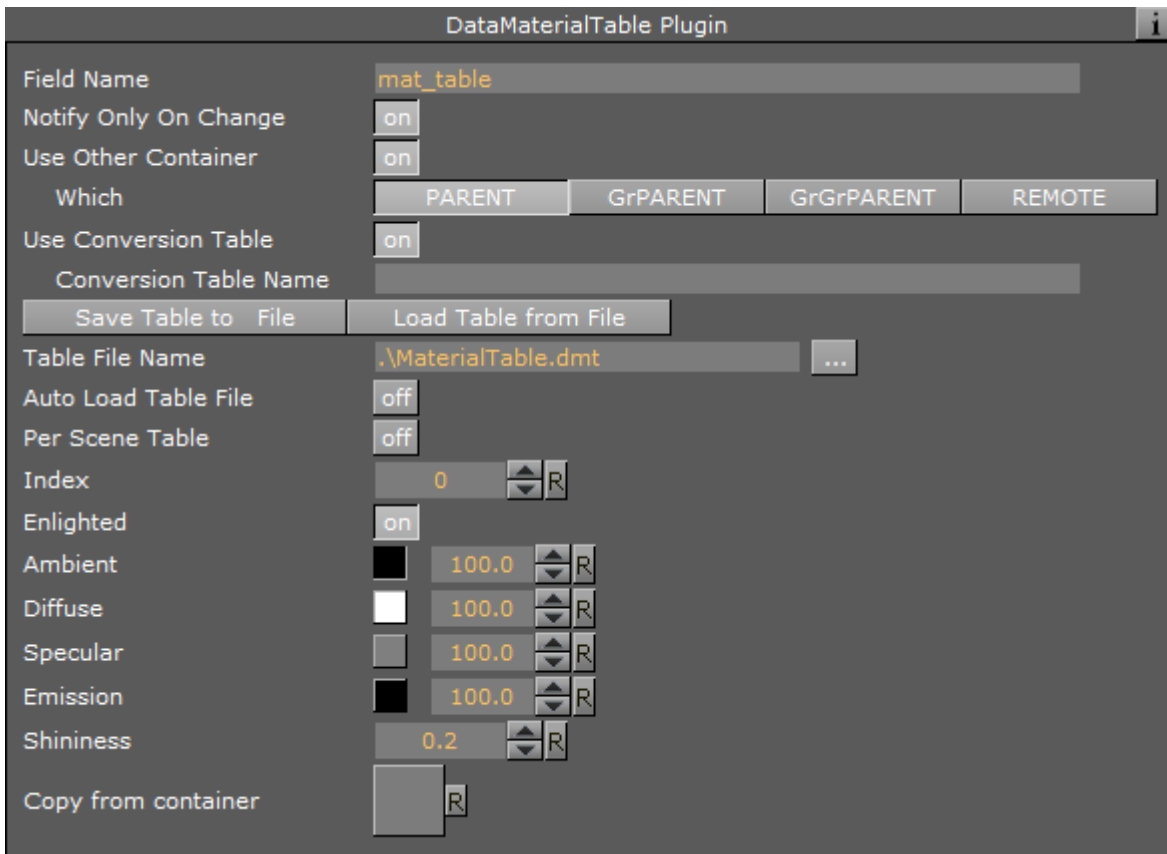
Unique Parameters

- **Save Table to File:** Saves table data to the defined file (in the Table File Name parameter) when pressed.
- **Load Table from File:** Loads table data to the defined file (in the Table File Name parameter) when pressed.
- **Table File Name:** Defines the name of a file for storing/reading the material table. Suffix for DataMaterialTable files is `.dmt`.
- **Auto Load Table File:** Loads the color table file during scene load when set to `On`. When set to `Off`, the scene loads without reloading the color table file.
- **Per Scene Table:** Loads and stores the color table used in the scene in a separate area from the common color table when set to `On`. When set `Off`, the scene uses a common color table defined in the table file.

> ⚠ **Note:** When using the common color table, every time the color table is loaded it overwrites the existing table. When using the Per Scene Table option, a color table is created for each scene.

- **Index:** Defines the entry in the material table (index) to be edited. Index is zero based, i.e. the first entry is `0`, the second is `1`, and so on.
- **Enlighted:** Defines whether the material in the entry index is lit.
- **Ambient:** Defines the ambient component of the material in the entry index of the table.

- **Diffuse:** Defines the diffuse component of the material defined in the current index of the table.
- **Specular:** Defines the specular component of the material defined in the current index of the table.
- **Emission:** Defines the emission component of the material defined in the current index of the table.
- **Shininess:** Defines the shininess component of the material defined in the current index of the table.
- **Copy from container:** Allows copying a material from a given container to the material table. The material parameters are set when the container is dragged to the container place holder, in the plugin editor. The material table is not changed if the material of the container is changed.

6.1.4   DataMouseSensor

The DataMouseSensor plugin receives mouse events and keyboard events from Viz and stores the events in special DataPool Variables, making this information accessible to other DataPool plugins.

The variables that it creates are:

- **MPOS:** Returns the mouse position when a button is clicked. If a button is pressed while moving the cursor MPOS updates dynamically. When the button is released the mouse position value is stored in MPOS. The position is displayed in X Y values.
- **MBUT:** Returns the number of the last clicked mouse button (1=left), whether the button is currently pressed or released, and the X Y values like in MPOS.
- **MSELCONT:** Returns the ID of the selected container.
- **KEY:** Returns the ASCII code of the last pressed key on the keyboard and whether it is currently pressed (pressed=1, released=0).

> ⚠ **Note:** Not all keyboard buttons are displayed when using the KEY variable.

- **MONOFF:** Indicates whether a mouse button is currently clicked or released.
- **MBUTTON:** Returns the number of the last clicked mouse button. 1=left button, 2=center button, 3=right button.

The DataMouseSensor is a scene plugin and it should be used in scenes that use mouse/keyboard DataPool plugins. The DataMouseSensor plugin has no parameters.

6.1.5   DataPool

The DataPool Plugin is a scene plugin that manages the DataPool mechanism and data transfer in the scene level (i.e. manages the DataFields/values/registered plugins table).
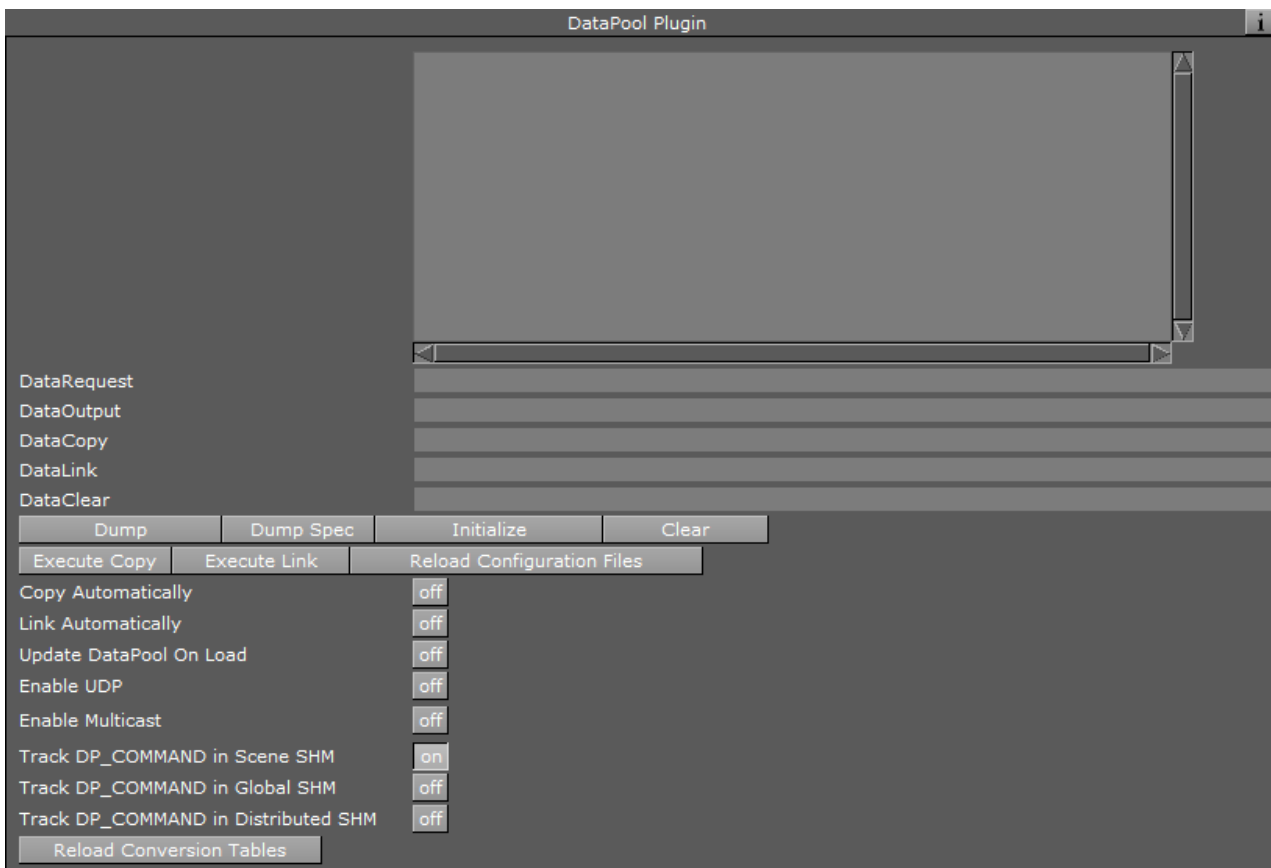
Unique Parameters

- **Data:** Allows the user or external software to define new DataFields and assign data to DataFields. The field content is saved with the scene when the scene is saved in Viz. Data is a text parameter. All commands/assignments should be semicolon separated. For example:

  ```
  A=5; B=3; aa[0..2]=1,3,5;  Stocks[1]={Name=TEST, Value=4.5};
  ```

- **Data1:** Sends information from external applications to the DataPool mechanism. The *Data1* field has a larger size and can accept larger information blocks than the Data parameter. This parameter is hidden. The data that is sent is not saved with the scene.

- **DataRequest:** Requests data from the DataPool (*DataRequest* is a text parameter). The user or external software can query DataFields by name using this parameter. The format of the query is as follows:

  ```
  fieldname[indexes-range]; fieldname[indexes-range]….
  ```

  The result of the query appears in the *DataOutput* parameter.

- **DataOutput:** Contains the data returned as a result of a query entered in *DataRequest* (*DataOutput* is a string parameter). The user or external software can use the returned information for manipulating the graphics. For example: External usage of *DataRequest* and *DataOutput*.

```
Enter data to the vectors VALUE and AAA:
0 RENDERER*FUNCTION*DataPool*Data SET VALUE[0-3]=V0, V1,  V2,V3;
AAA[0-5]=A0, A1, A2, A3, A4, A5;

Request the last two items of VALUE and the third value of AAA:
0 RENDERER*FUNCTION*DataPool*DataRequest SET VALUE[2-3];  AAA[3];

Read the requested information:
0 RENDERER*FUNCTION*DataPool*DataOutput GET

Viz replies:
0 VALUE[2-3]=A2, A3; AAA[3]=A3;
```

- **DataCopy:** Copies data from one DataPool variable or structure to another. For example:
    - a=b; is assigned the value of b to a.
      Or, if Stock1 and Stock2 is of Stocks type then:
    - Stock2=Stock1; assigns all the values from Stock1 to the values of Stock2.

    > ⚠ **Note:** The copy action does not take place unless the Copy Automatically parameter is set On, or the **Execute Copy** button is pressed.

- **DataLink:** Links one DataPool variable or structure to another. For example:
    - a=b; links the value of a to b.
      or, if Stock1 and Stock2 is of Stocks type then:
    - Stock2->Stock1; links Stock2 to Stock1.

    > ⚠ **Note:** The link action does not take place unless the Link Automatically parameter is set On, or the **Execute Link** button is pressed.

- **Dump:** Triggers a printout of all DataPool variables, structures and their values to the Viz console and to the file *datapool_dump.txt* located in *\ProgramData\Vizrt\viz3*.
- **Dump Spec:** Triggers a printout of all DataPool variables and structures to the Viz console.
- **Initialize:** Triggers a rebuild of the DataPool hierarchy in the scene and the assignment of values to the DataPool variables.
- **Execute Copy:** Triggers an immediate copy of the variables or structures defined in the *DataCopy* field.
- **Execute Link:** Triggers an immediate link of the variables or structures defined in the *DataLink* field.
- **Reload Configuration Files:** Allows reloading all the *.dp files.
- **Reload Conversion Tables:** Allows reloading all the conversion tables.
- **Copy Automatically:** Defines if the copy of the values, defined in the *DataCopy* field, is triggered automatically when the *DataCopy* parameter is changed. When set OFF the copy operation does not happen until the **Execute Copy** button is pressed. When set to On, every time the *DataCopy* parameter changes or any of the copied DataPool variables defined in the *DataCopy* field is modified, DataPool copies the values as defined in the *DataCopy* parameter.

- **Link Automatically:** Defines if the link of the values, defined in the *DataLink* field, is triggered automatically when the *DataLink* parameter is changed. When set to `Off` the copy operation does not happen until the **Execute Link** button is pressed. When set to `On`, every time the *DataLink* parameter changes or the linked DataPool variables are modified, DataPool executes the link as defined in the *DataLink* parameter.
- **Update DataPool On Load:** Loads and initializes the data stored in the Data parameter when set to `On`.
- **Enable UDP:** Enables external applications to control the DataPool via UDP when set to `On`. When set to `On` additional parameters are enabled.
- **Show UDP Messages:** Defines whether to display incoming UDP messages in the Viz console.
- **UDP Host Name:** Defines the name of the machine that the UDP messages should arrive from.
- **UDP Port:** Defines the port number that DataPool listens for UDP messages.
- **Enable Multicast:** Enables external applications to control the DataPool via MULTICAST when set to `On`. When set to `On` additional parameters are enabled.
- **Show Multicast Messages:** Defines whether to display incoming MULTICAST messages in the Viz console.
- **Multicast Port:** Defines the port number that DataPool listens for MULTICAST messages.
- **Multicast Address:** Defines the address of the machine sending the MULTICAST messages.
- **Track DP_COMMAND in Scene SHM:** Tracks a reserved shared memory key `DP_COMMAND` in the scene shared memory when set to `On`. When the key is changed its content is parsed and interpreted as one of the following commands:
    a. If `DP_COPY` prefix is used, the remaining string is used as a DataPool copy command, copying one or multiple DataPool fields.
    b. If `DP_LINK` prefix is used, the remaining string is used as a DataPool link command, linking one or multiple DataPool fields.
    c. If `DP_SET` prefix is used, the remaining string is used to set the value of DataPool fields.
    d. When none of the above options are detected, the content of `DP_COMMAND` is used to set the value of DataPool fields.
- **Track DP_COMMAND in Global SHM:** Tracks a reserved shared memory key `DP_COMMAND` in the scene shared memory when set to `On`. The same as the above section, regarding Viz 3 Global shared memory.
- **Track DP_COMMAND in Distributed SHM:** Tracks a reserved shared memory key `DP_COMMAND` in the scene shared memory when set to `On`. The same as the above section, regarding Viz 3 Distributed shared memory.

6.2 Container Plugins

The container plugins are listed in this section in alphabetic order.

- Data3DObject
- DataAction
- DataActionTable
- DataAlpha
- DataAnim

- DataArray
- DataArrow
- DataCamera
- DataCenter
- DataClick
- DataCondition
- DataCountdown
- DataCounter
- DataDirector
- DataDispatcher
- DataDrawMask
- DataFeedback
- DataGeom
- DataGraph
- DataGraphPoint
- DataHyperlink
- DataImage
- DataInRange
- DataKey
- DataKeyFrame
- DataKeyFrame2
- DataKeyText
- DataKeyTime
- DataLookup
- DataLUImage
- DataManipulate
- DataMaterial
- DataMaterialGradient
- DataMaterialIndex
- DataMath
- DataMathObject
- DataMinMax
- DataMouseAction
- DataMousePosition
- DataMultiParam
- DataNumber
- DataObject
- DataObjectTracker
- DataParameter
- DataParamTracker
- DataPosition
- DataReader
- DataRotation
- DataScale
- DataScreen

- DataScript
- DataSelector
- DataSHM
- DataSHMTracker
- DataStructure
- DataSwitch
- DataSystem
- DataTable
- DataText
- DataTextKerning
- DataTexture
- DataTime
- DataTimer
- DataViz3Script
- DataWPosition

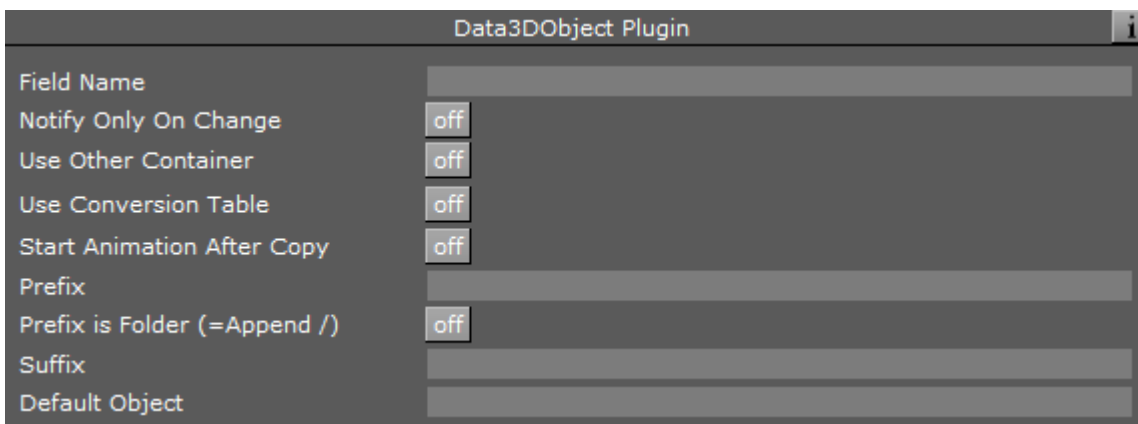## 6.2.1 Data3DObject

Data3DObject receives a name of an object from the Viz object library, and it loads it to the container it is placed on.

> ⚠ **Note:** Built in objects cannot be used.

Prefix is used to add a Viz object path to a folder where the objects are stored.
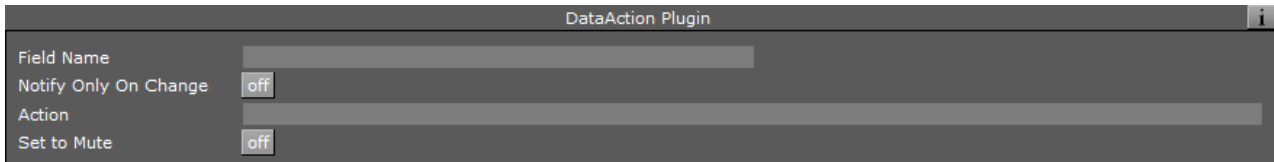

For example: If the FieldName is *AAA* and the object is *spaceship* then the command is:

```
AAA=spaceship;
```

6.2.2   DataAction

DataAction executes an action (i.e. Viz commands or DataPool commands) when the DataField registered to it changes.

DataAction Plugin		i
Field Name		
Notify Only On Change	off	
Action		
Set to Mute	off	

Unique Parameters

· **Action:** Specifies the action to be invoked. If the action string begins with `0`, then the command is handled as a Viz command, otherwise it is considered to be a DataPool command. The action can contain more than one command to be called. The commands must be separated by semicolons (;). Expressions can be used in the action utilizing the special DataPool variables.

· **Set to Mute:** Specifies if the action is triggered when the data field changes.

6.2.3   DataActionTable

DataActionTable allows defining a table of actions (as described in the DataAction plugin page). Each of these actions is identified by a value of the FieldName variable. When DataActionTable receives data (a value of the FieldName variable) it compares the data to each of the values. If the data matches one of the values, the plugin invokes the corresponding action.
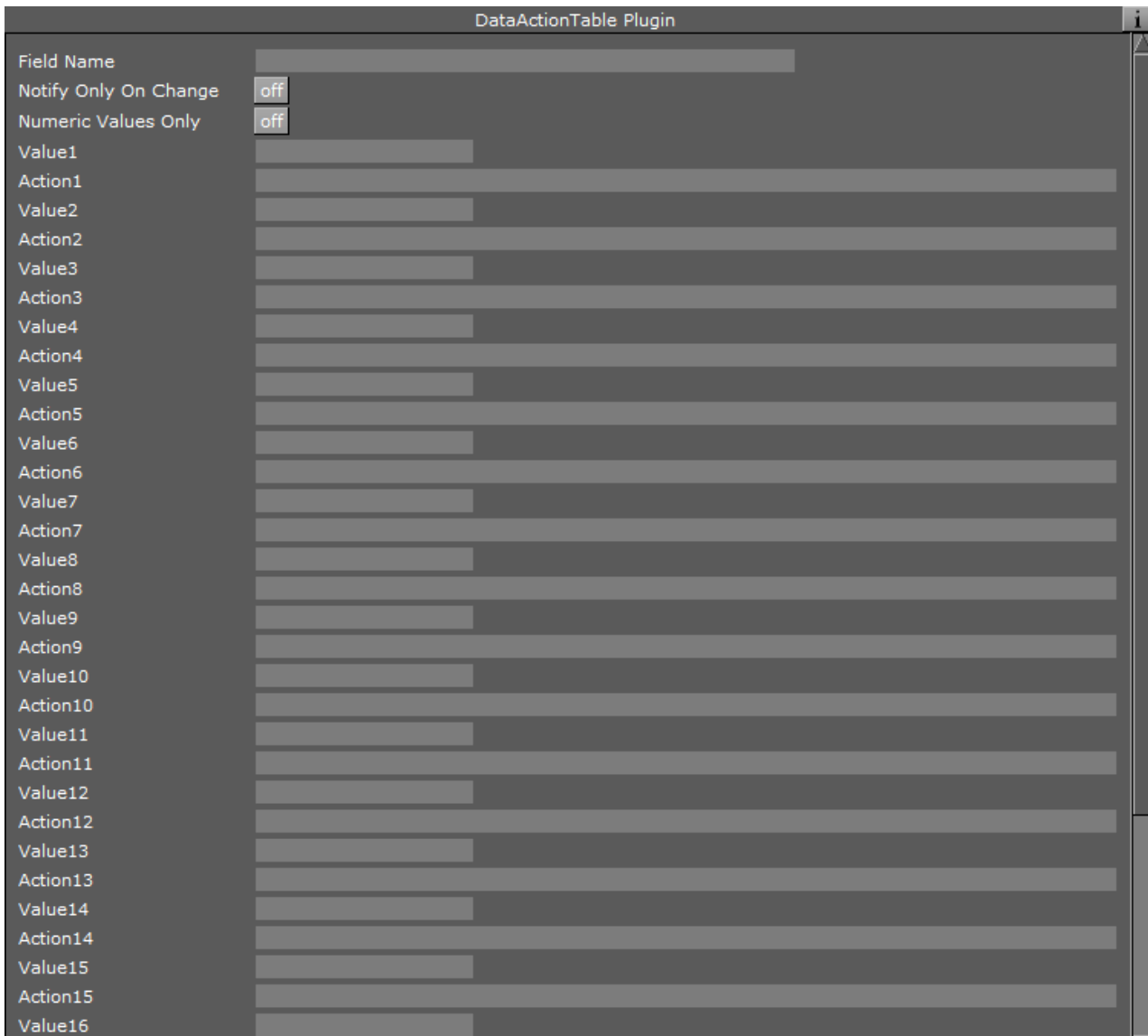
Unique Parameters

- **Numeric Values Only:** Enforces the plugin to use only numeric values received as the Field Name value, and uses only numeric values in the `Value<i>` parameters, when set to `On`.
- **Value<i>:** Specifies the value of the FieldName variable that invokes the Action specified in `Action<i>`.
- **Action<i>:** Specifies the action to call in the case the data matches `Value<i>`.
- **Default Action:** Specifies the action to invoke if the data doesn't match any of the specified values.

6.2.4   DataAlpha

☐

DataAlpha controls the value of the Alpha function plugin. Values are between `0` and `100` (`0`=transparent, `100`=opaque).

> ⚠ **Note:** Alpha function plugin must be assigned on the controlled container.
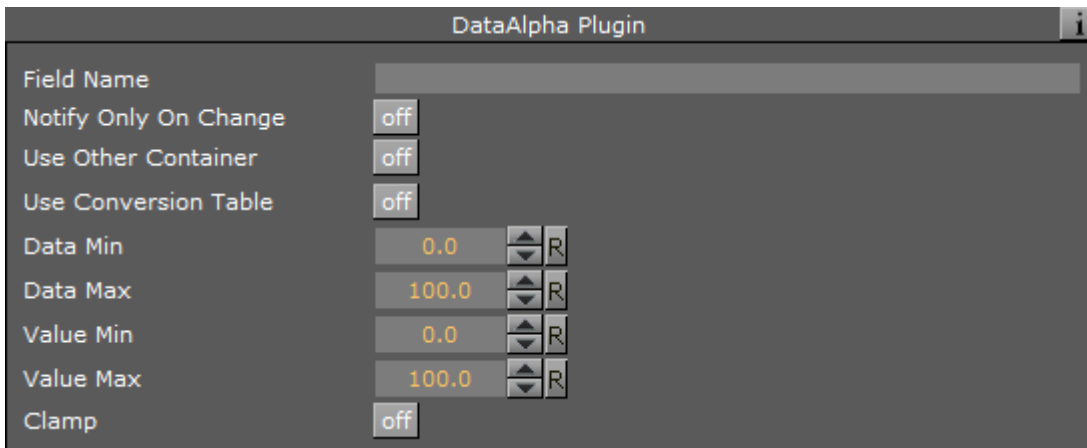
6.2.5   DataAnim

☐

DataAnim animates a value and sends the data of this value, every field, to the DataField that is defined in the Output Field Name. This plugin is different from other DataPool plugins since it is not triggered by a change of a DataField. It is activated from the animation director it is located in, and plays a predefined animation. This plugin acts as a data generator for the specified output DataField.
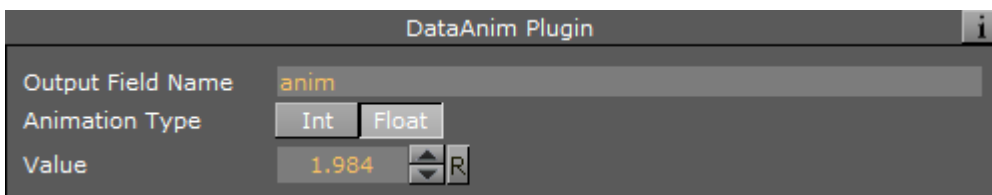
Unique Parameters

- **Output Field Name:** Defines the name of the DataField that receives the animated value of the *Value* parameter every video field.
- **Animation Type:** Defines the type of value that is sent to the Output Field Name. Value types are integer or float.
- **Value:** Generates and sends the data (the animated parameter of the plugin) to the DataField defined in Output Field Name.

Example

Define the Output Filed Name as `AAA`, and create an animation of the Value parameter from `0` to `10`. When running the animation, `AAA` is changed every field as the animation progresses from `0` to `10`. All DataPool plugins registered to the DataField `AAA` is then triggered every field and set to the current value of the Value parameter.

6.2.6   DataArray

DataArray enables a simpler way of controlling a complex structure of objects that are defined in the `config.dp` file (See DataObject for detailed information about defining object structures and using the `config.dp` file). The plugin defines the size of the array (the number of its child objects) and manages the data flowing to the objects.
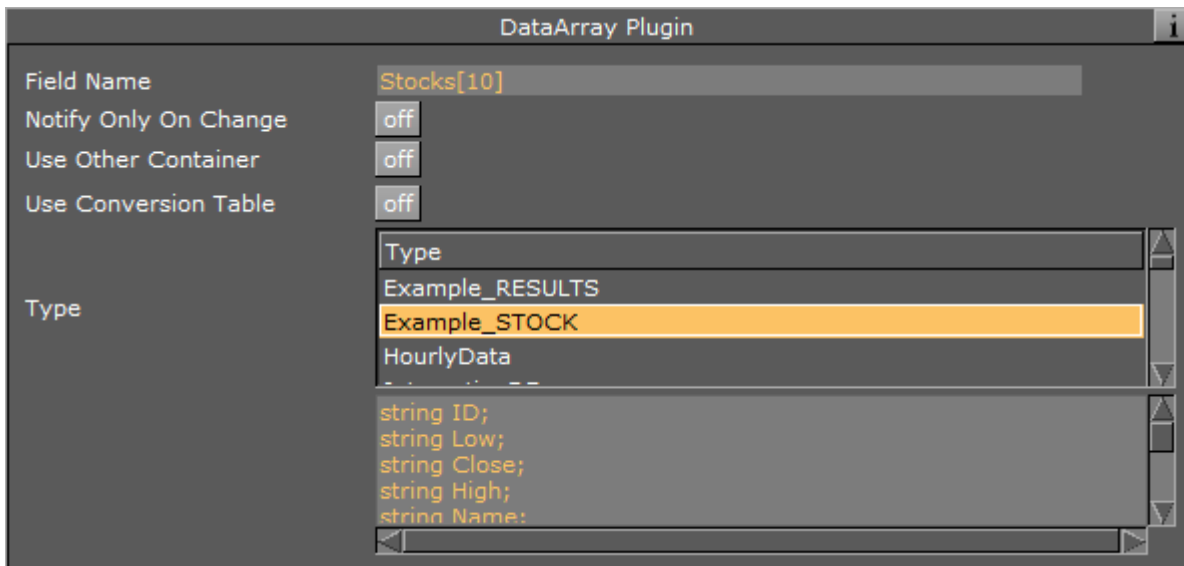
Unique Parameters

· **Field Name:** Defines the name and size of the array.
· **Type:** Defines the type of the array objects. The list of types is created from the types defined in the configuration files (`.dp` files).

Example

The Field Name `Stocks[32]` defines an array named Stocks that has 32 objects of the type Stock selected in the Type parameter.

6.2.7 DataArrow

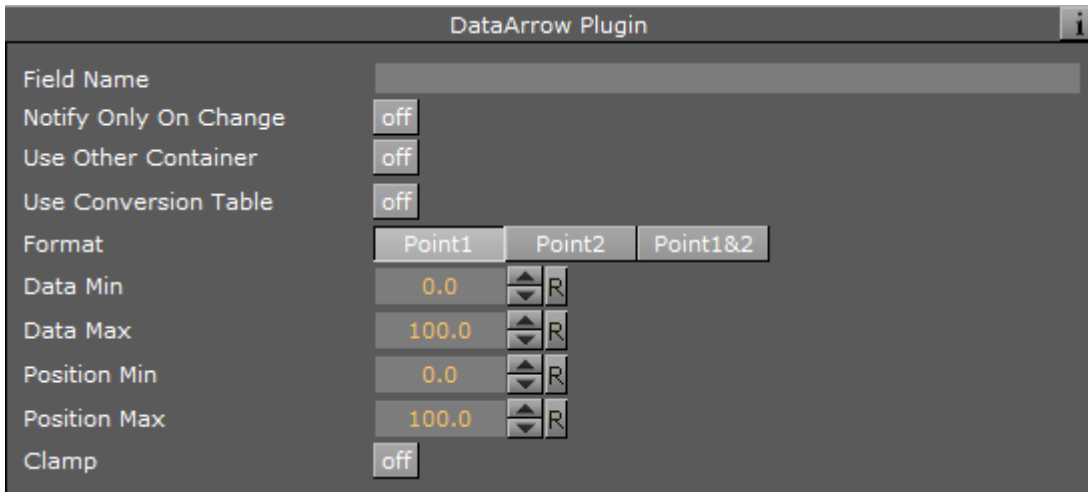The DataArrow plugin works specifically on the Arrow geometry plugin in Viz. It controls the start and end points of the arrow object, enabling incoming data to control the arrow object.

Unique Parameters

- **Format:** Specifies the format of the expected data. The plugin processes the data according to the selected format. The options are *Point1* (start point), *Point2* (end point) or *Point1&2* (both end points).

6.2.8 DataCamera

The DataCamera plugin controls camera position and target according to the incoming data. If the container that the plugin controls has child containers, then the specified camera locks its position to the first child container and the destination of the camera is locked to the second child container. In this case the data should be of the following format:

```
DATA=<camera> <x position> <y position> <z position> <x target> <y target> <z
target>;
```

> ⚠ **Note:** Position and target values are used as offsets from the child containers location.

If the container doesn't have any child containers then the data should be of the following format:

```
DATA=<camera> <x position> <y position> <z position> <pan> <tilt> <twist>;
```

> ⚠️ **Note:** Camera position change is relative to the container that the plugin works on.

The plugin works in two different modes: First, the data contains both the camera ID, the position and target of the camera and second, the data contains just the camera ID and the position and the target of the camera are calculated from the center of the first two children of the container. In the second mode, the container must have two children. The center of the first child determines the center of the camera and the center of the second child determines the target of the camera.
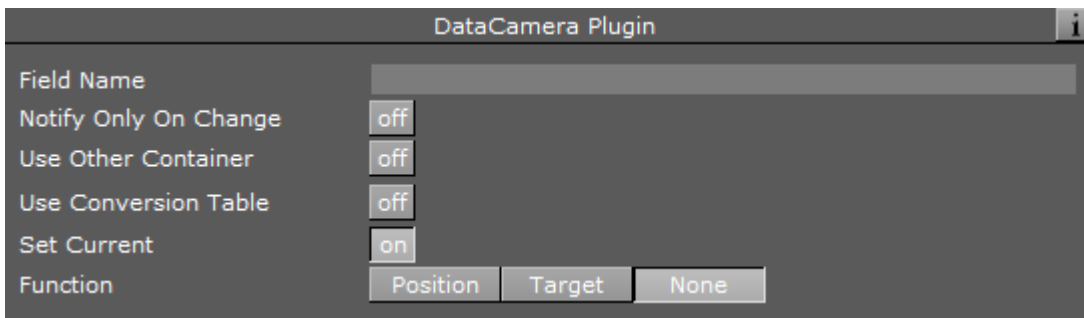
Unique Parameters

- **Set Current:** Defines whether a camera switch in the render window occurs if the incoming data refers to a different camera than the current camera. If set to `On`, then when incoming data refers to a camera other than the current camera selected in the render window, the camera switches to the camera defined in the incoming data. When set to `Off`, the camera defined in the incoming data is moved, but the current camera remains the same.
- **Function:** This parameter is inactive. The parameter is kept for backwards compatibility and should not be used.

6.2.9   DataCenter

DataCenter changes the axis of the container according to the received data.
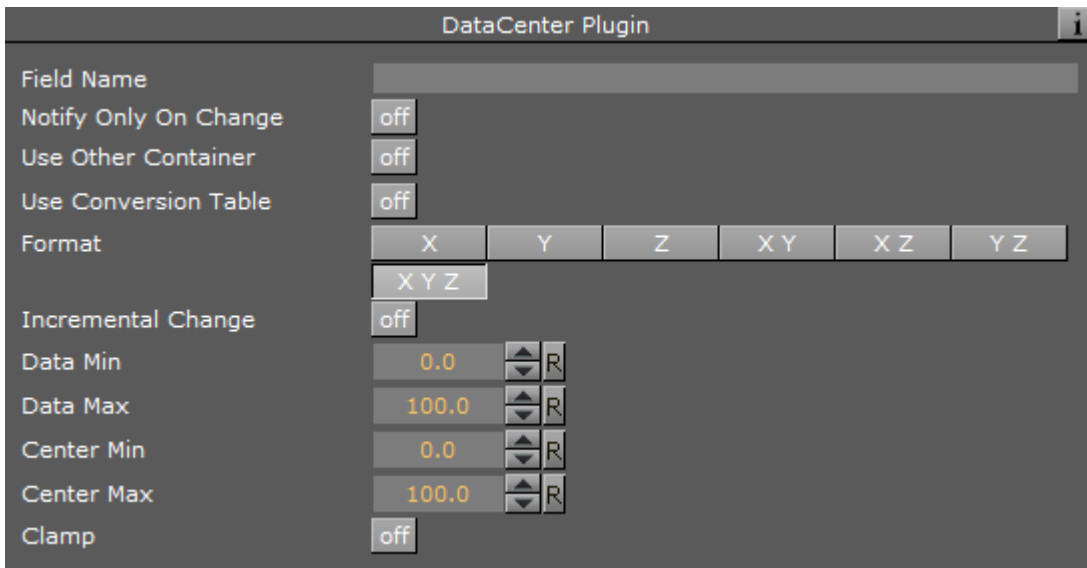
Unique Parameters

- **Format:** Specifies the format of the incoming data. Format options specify the axis which the data relates to.
- **Incremental Change:** Adds incoming data to the current value of the data when set to `On`. When set `Off`, the data replaces the existing value of the data field.

Example

If Field Name is `VALUE` and the format is `X`, the data sent is `VALUE=10.5`. The container moves the X axis to a position of `10.5`.

6.2.10   DataClick

The DataClick plugin triggers an action when the mouse is clicked within the drawn area of the container it is assigned to or on any of its child containers.
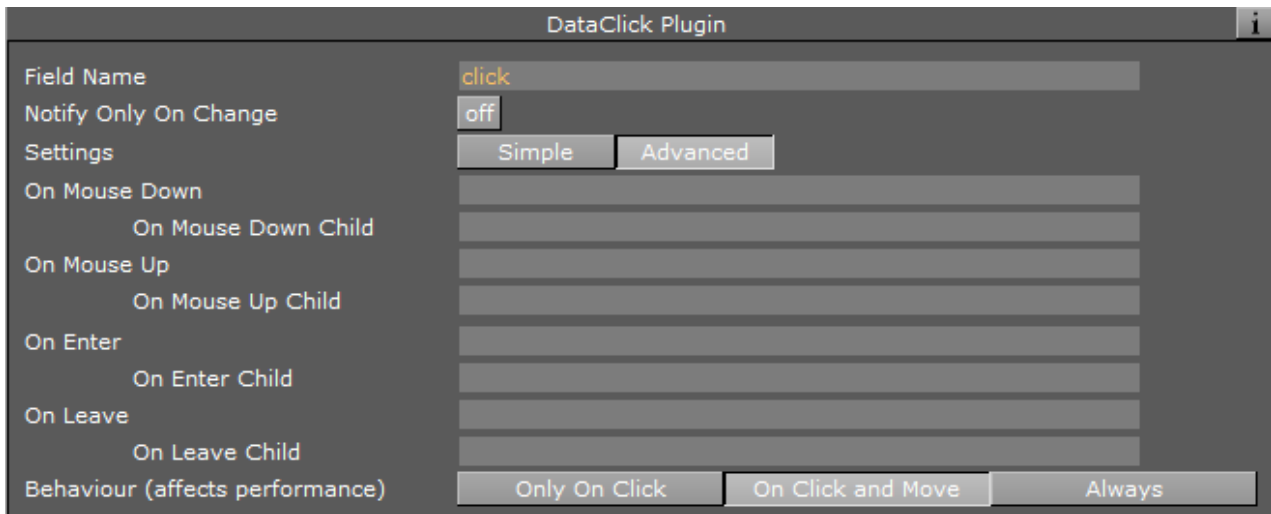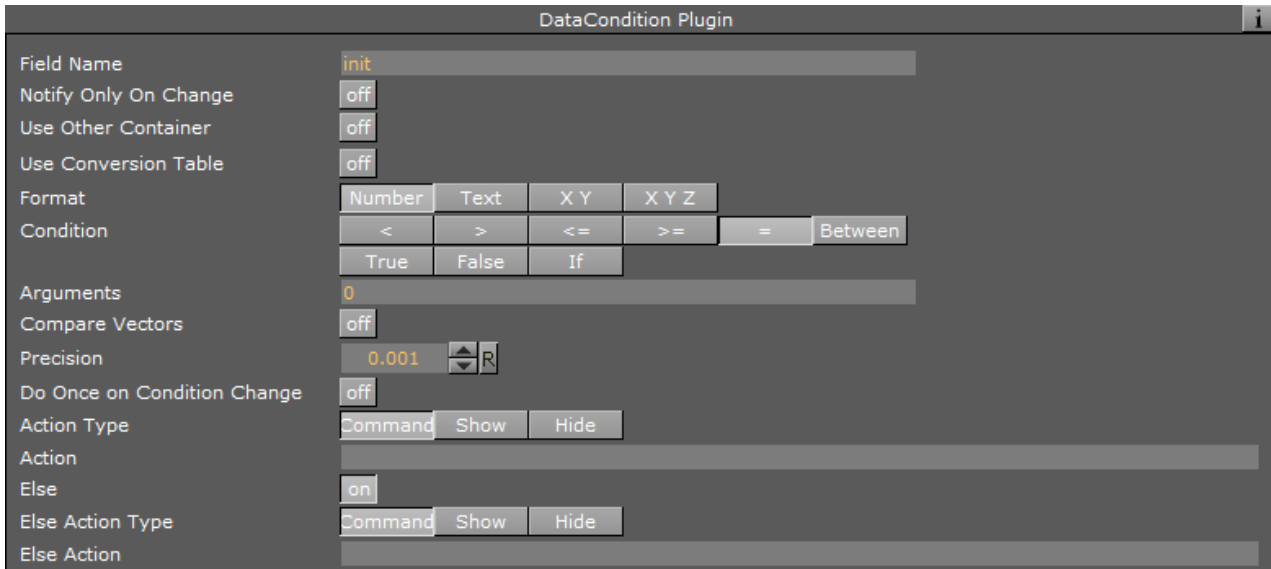
Unique Parameters

- **Settings:** Defines the displayed editor type. When set to **Advanced**, every click triggers two events: mouse down (mouse button pressed) and mouse up (mouse button released). When set to **Simple**, only mouse click is used.
- **On Click:** Triggers a DataPool action when the mouse is clicked within the container's drawn area and any of its child containers.
- **On Click Child:** Triggers a DataPool action when the mouse is clicked within the drawn area of a child container under the container affected by the DataClick plugin.
- **On Mouse Down:** Triggers a DataPool action when the mouse button is pressed within the container's drawn area and any of its child containers.
- **On Mouse Down Child:** Triggers a DataPool action when the mouse button is pressed within the drawn area of a child container under the container affected by the DataClick plugin.
- **On Mouse Up:** Triggers a DataPool action when the mouse button is released within the container's drawn area and any of its child containers.
- **On Mouse Down Child:** Triggers a DataPool action when the mouse button is released within the drawn area of a child container under the container affected by the DataClick plugin.
- **On Enter:** Triggers a DataPool action when the mouse enters the container's drawn area and any of its child containers.
- **On Enter Child:** Triggers a DataPool action when the mouse enters the drawn area of a child container under the container affected by the DataClick plugin.
- **On Leave:** Triggers a DataPool action when the mouse leaves the container's drawn area.
- **On Leave Child:** Triggers a DataPool action when the mouse leaves the drawn area of a child container under the container affected by the DataClick plugin.
- **Behavior:** Defines in what way the mouse events trigger the actions or cause the DataField to change. When set to **On Click**, only the on click events are enabled. When set to **On Click and Move** additional action (On Enter and On Leave) fields are enabled.

6.2.11 DataCondition

☐
DataCondition evaluates the value of the DataField it is registered to using a defined comparison function and executes an action accordingly.

Unique Parameters

· **Format:** Defines the way incoming data is handled. Data can be used as a number, string, a set of X Y values or a set of X Y Z values. The following table shows how the different formats are evaluated. Cells filled with EVAL are evaluated by the selected operator. Cells filled with TRUE always return true. Cells filled with FALSE always return false. Other text in a cell shows the behavior of the chosen operator.

> ⚠ **Note:** The X Y and X Y Z formats only support equal to (=) and not equal to (!=).

	<	>	<=	>=	=
Number	EVAL	EVAL	EVAL	EVAL	EVAL
Text	EVAL	EVAL	EVAL	EVAL	EVAL
X Y	FALSE	!=	=	TRUE	EVAL
X Y Z	FALSE	!=	=	TRUE	EVAL

· **Number:** Performs a simple numeric evaluation.
· **Text:** Performs an alphanumeric evaluation.
· **X Y:** Calculates the sum of the absolute value of the difference between the first and second values of the arguments and compares it to the precision value multiplied by two: |x1 - x2| + |y1 - y2| < 2 * precision.

- **X Y Z:** Calculates the sum of the absolute value of the difference between the first, the second and the third values of the arguments and compares it to the precision value multiplied by three: |x1 - x2| + |y1 - y2| + |z1 - z2| < 3 * precision.
- **Condition:** Specifies the evaluation function to apply. Evaluation is done left to right (incoming data is placed at the left side of the comparison operator). `True` always runs the action and `False` always runs the *else* action (if enabled). The *If* option evaluates the DataField value: i.e. all values other than zero runs the action, zero runs the *else* action. `Between` checks if the value is greater or equal (>=) than the first and lower (<) than the second argument.
- **Arguments:** Defines the expression that is compared to the incoming data. An argument can be a constant (i.e. a fixed number/text) or a data field value. To specify a data field value as an argument, use the following syntax: `$(DataFieldName)`.
- **Compare Vectors:** Defines incoming data as a vector (an array of DataFields with definition of the number of items) and the argument expression is taken to be a vector of data when set to `On`. The comparison is then done item by item and the action is executed accordingly on each of the children. If this parameter is `Off` then each item in the incoming data is compared against one single argument.
- **Precision:** Affects the evaluation of X Y and X Y Z formats only. It defines the margin in which the evaluation results as true (see X Y or X Y Z format explanation).
- **Do Once On Change:** Triggers an Else action. When set `ON`, only if the condition result was changed, i.e. if it was changed from true to false or vice versa, the action (or Else action) is triggered. When set `Off`, every time the condition is checked the action (or Else action) is triggered accordingly.
- **Action:** Specifies the action to invoke. If the action string begins with `0`, then the command is handled as a Viz command and sent directly to Viz Engine, otherwise it is considered to be a DataPool command. The action can contain more than one command to be called. The commands must be separated by semicolons (;). Expressions can be used in the action utilizing the special DataPool variables.
- **Else:** Enables the Else action when set to `On`. When an else action is defined it is executed if the evaluation result is false. When set `Off`, the Else action field is disabled and no action is executed when the evaluation result is false.
- **Else Action:** Defines the action to execute if the evaluation result is false.

6.2.12 DataCountdown

☐

DataCountdown is used to count down or count up to a defined target time. Target time is an absolute time (future or past).

Unique Parameters

- **DP variables Prefix:** Variable prefix is added to a set of fixed names, based on time scale, for different time variables. The prefix is used to distinguish one set of time parameters from the other. The default prefix is CD1. For example, if Prefix is set to DDay, the corresponding variables are:

```
DDay_DAYS
DDay_HOURS
DDay_HOURS_TOTAL
DDay_MINS
DDay_MINS_TOTAL
DDay_SECS
DDay_SECS_TOTAL
DDay_SIGN (i.e + for future and: for past)
```

- **Target Day in month:** Defines the countdown target day. Values are 1-31.
- **Target Month:** Defines the countdown target month. Values are 1-12.
- **Target Year:** Defines the countdown target year. Values are 1970-2050.
- **Target Hour:** Defines the countdown target hour. Values are 0-23.
- **Target Minute:** Defines the countdown target year. Values are 0-59.
- **Target Second:** Defines the countdown target year. Values are 0-59.
- **Leading Zero All:** Adds a leading zero to all the single digit numbers (0-9) when set to On.
- **Leading Zero Minutes, Seconds:** Adds a leading zero to the single digit numbers (0-9) of the minutes values and the seconds values when set to On. Others are displayed as a single digit.
- **Action On Time:** Defines an action that is executed when the counter reaches the target time. If the action string begins with 0 then the command is handled as a Viz command, otherwise it is considered to be a DataPool command. The action can contain more than one command to be called. The commands must be separated by semicolons (;). Expressions can be used in the action utilizing the special DataPool variables.

- **Action On T-offset:** Defines the action that is executed when the counter reaches an offset point from the target time.
- **Offset (Seconds):** Defines the offset from the target time that triggers the action on *t*-offset (in seconds).
- **Set Target to NOW:** Sets the current system time as the target time when pressed.

6.2.13 DataCounter

DataCounter plugin increases/decreases its value every time the DataField changes (if set to begin counting at 1 with an incremental value of 1, the plugin counts the number of times that the DataField changed). The plugin starts counting at the *From* parameter value, adding the *Step* parameter value to counter every time the DataField defined in Field Name changes, until the value *To* is reached. When the *To* value is reached, the counter starts over again from the beginning.

Unique Parameters

- **From:** Defines the initial value of the counter.
- **To:** Defines the ending value of the counter.
- **Step:** Defines the value to increment.
- **Output Field:** Defines a DataPool variable that the counter value is assigned to.
- **Output Scope:** Defines whether the DataPool variable defined in the output field is a global variable or a local variable.

6.2.14 DataDirector

DataDirector plugin enables the control of an animation director by assigning stage commands to the DataField it is registered to.
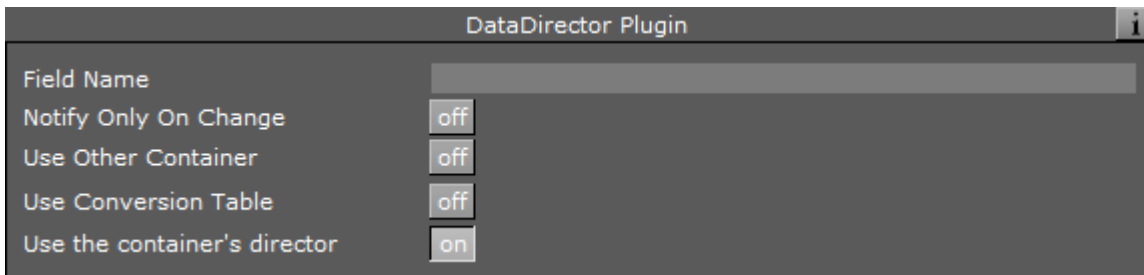
Unique Parameters

- **Use the container's director:** Defines if the plugin controls the director for the animation in a container.

  > ⚠ **Note:** If Use Remote Container is used, then the animation director in the remote container is triggered.

  If none is enabled then the default director is controlled.

Example

To run the animation, send the command: `Director=START;`.

6.2.15 DataDispatcher

Dispatcher enables data transfer between plugins. DataDispatcher calls the dispatcher of a plugin in the controlled container, and transfers data via DataPool variables.
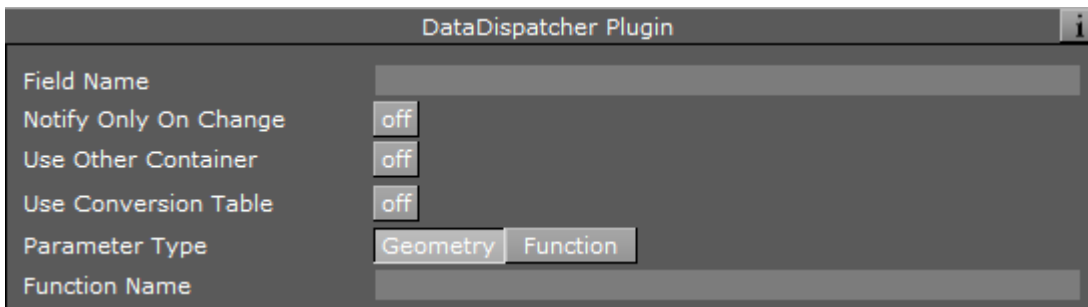
Unique Parameters

- **Parameter Type:** Defines destination plugin type. Data is formatted accordingly.
- **Function Name:** Defines the name of the plugin whose dispatcher is to be called.

Data Format

The data should arrive in the following format:

```
<dispatcher message ID> <dispatcher message body>
```

6.2.16 DataDrawMask

DataDrawMask plugin is used when using viewports (multiple camera windows in Viz render window). The plugin defines the mask type of the container it controls and the way it is rendered in each of the viewports. The DataField (defined in Field Name) value contains this information.
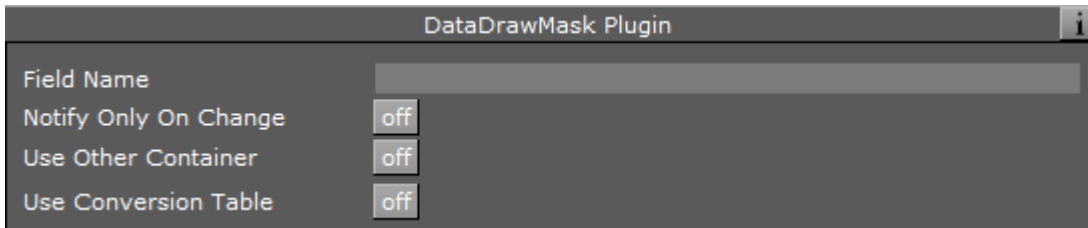

6.2.17 DataFeedback

DataFeedBack plugin sends information from the DataPool to a specific UDP socket in a remote host.
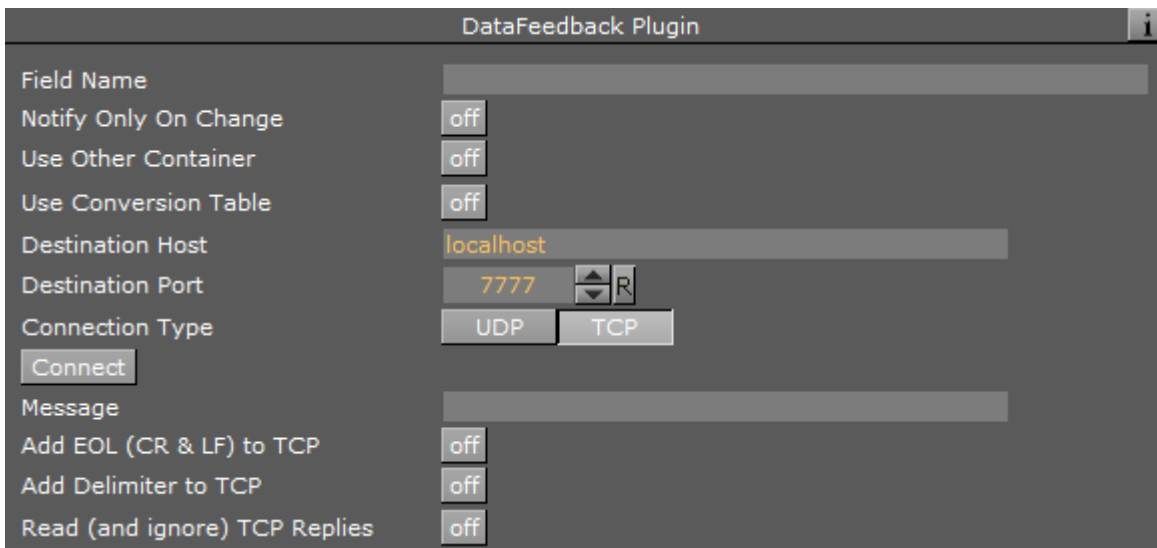
Unique Parameters

· **Destination Host:** Defines the hostname to which the data is sent to.
· **Destination Port:** Defines the socket number to which the data is sent to.
· **Connect:** Initializes the connection after the host and port are defined.
· **Connection Type:** Defines the connection type to open: UDP or TCP.
· **Message:** Defines the expression to send to the remote socket when the plugin is triggered.

- **Add EOL (CR &LF) to TCP:** Defines if new line and line feed characters to add to the TCP message.
- **Add Delimiter to TCP:** Defines if a delimiting character to add to the message.
- **Char Of TCP Delimiter:** Defines the character to use as a delimiter and add to the message.

6.2.18   DataGeom

The DataGeom plugin sets the geometry of the container it is assigned to according to the received data. A group with child geometry containers is defined in the scene tree and the DataField receives the index of the selected geometry (index is zero based). The same result can be achieved by using the Data3DObject plugin, but Data3DObject plugin accesses the disk when used. To avoid real time rendering problems in heavy scenes, it is better to use the DataGeom since it does not access the disk.
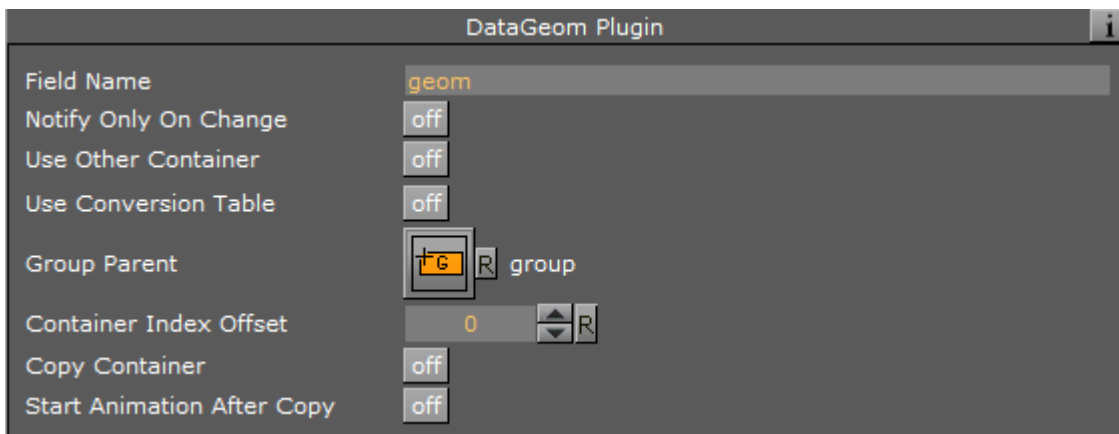
Unique Parameters

- **Group Parent:** Defines the parent container of the geometries in the scene tree. The container is dragged to the container area in the DataGeom editor.
- **Copy Container:** Defines if the geometry is copied as a child container to the DataGeom controlled container when sent.

Example

Given the following scene tree:

The DataPool assignment `Geom=0;` changes the geometry of the GEOM container to sphere, `Geom=1;` changes it to cube and so on.

> ⚠ **Note:** If an index value that does not exist is sent, then zero value is used (the first geometry under the group).

6.2.19   DataGraph

☐

The DataGraph plugin works on the Graph geometry plugin in Viz. It enables controlling the values of the graph points. Incoming data value of the DataField must contain all the point values according to the selected format, separated by spaces.
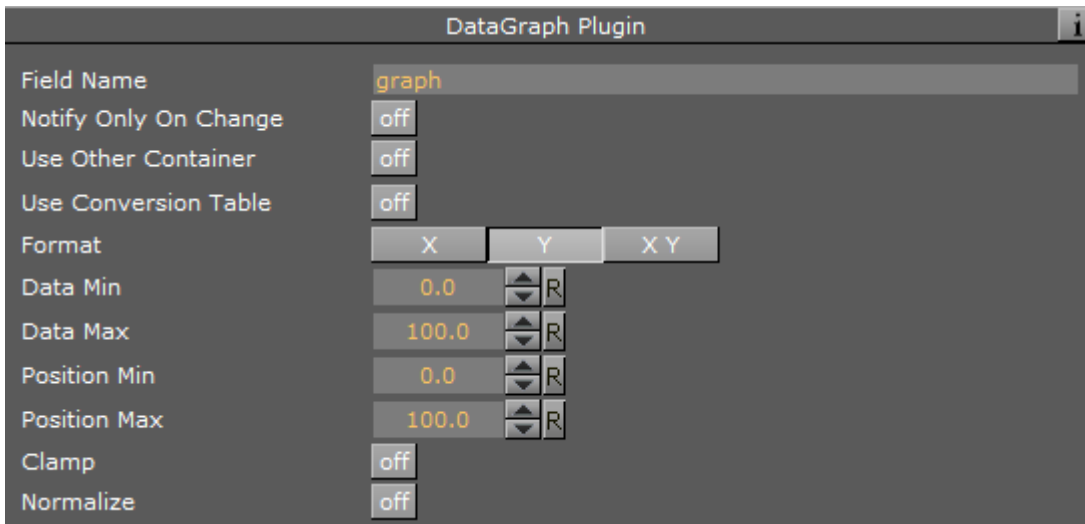
Unique Parameters

- **Format:** Defines the axis that is controlled by the plugin. If one axis is selected (X or Y) then the data should contain one value per point, if X Y is selected, then two values should be sent per point.
- **Normalize:** Forces the graph points to use the entire height range (the range between Position Min and Position Max parameter) according to the minimum and maximum values received when set to On. The minimum value is used as the Data Min value and the maximum value is used as the Data Max, using the entire graph range for the received values.

Example

If format is set to X Y, then for drawing four point of the graph, eight numbers should be sent:
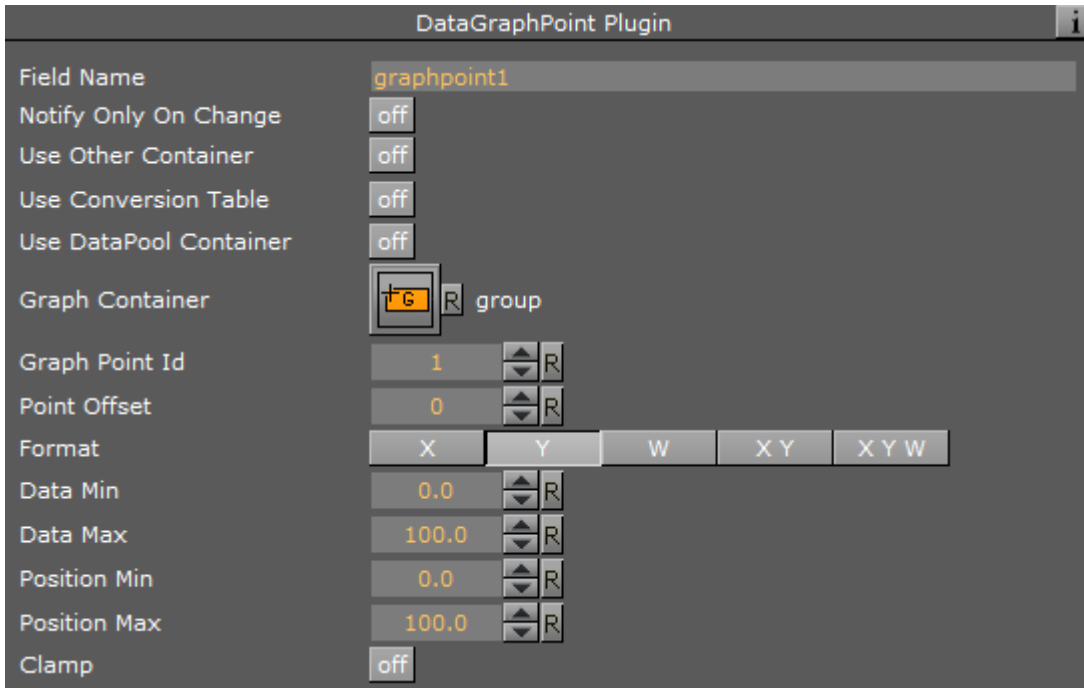
```
Abc=0 0 10 5 12 10 8 15;
```

- The first point is located at X=0, Y=0.
- The second point is located at X=10, Y=5.
- The third point is located at X=12, Y=10.
- The fourth point is located at X=8, Y=15.

6.2.20   DataGraphPoint

The DataGraphPoint enables to control values of a selected point in a Graph
object, 2DRibbon object and 2DPatch object.

Unique Parameters

- **Use DataPool Container:** Replaces the Graph Container parameter with the Container Field parameter when set to 0n. This enables a dynamic control of different graph objects defined in the Container Field parameter.
- **Container Field:** Defines a DataPool variable which contains a container name, replacing the Graph Container parameter placeholder. This parameter is used from external applications sending data to DataPool enabling the application to change the controlled graph object.
- **Graph container:** Defines the container to be controlled. Drag the geometry object from the Viz tree to the container area.

> ⚠ **Note:** Graph Container and Container Field parameters operate the same as the Use Remote Container parameter. The Use Remote Container parameter has no effect in the plugin.

- **Graph Point ID:** Defines the point to control. ID is the index of the point in the geometry plugin. ID 1 refers to the first point and so on.
- **Point Offset:** Defines an offset of the point index from the first point. The actual point ID is the sum of Graph Point ID and Point offset.

- **Format:** Defines the axis or width to be controlled by the plugin. If one axis is selected (X or Y) or if width is selected (W), then the data should contain one value. If X Y is selected, then two values should be sent. If X Y W are selected, three values should be sent.

6.2.21   DataHyperlink

DataHyperlink plugin receives mouse events and triggers actions accordingly. The actions are Viz and DataPool commands. This plugin gets the mouse events from Viz and it does not require the DataMouseSensor plugin added to the scene. When a DataHyperlink plugin is used in a scene, a set of predefined DataPool variables is created and updated on every change in the scene. These variables contain information about the mouse cursor location and other mouse events:

- **MX:** Defines cursor location on the X axis.
- **MY:** Defines cursor location on the Y axis.
- **MXY:** Defines cursor location on the X axis and Y axis.
- **MXYZ:** Defines cursor location on the X axis, Y axis and Z axis.
- **MOUSE_DRAG:** Indicates whether a mouse drag is performed.
- **MOUSE_CLICK:** Indicates whether a mouse button is currently pressed.
- **DRAGGED_CONT:** Returns the ID of the currently dragged container
- **DROPPED_CONT:** Returns the ID of the container that was dropped on the container hosting the DataHyperlink plugin
- **DROP_TYPE:** Defines the type of object that was dropped on the container: No object, geometry, image.

> ⚠ **Note:** When using DataHyperlink in a scene and no other data is sent externally, adding the DataPool scene plugin is unnecessary.

Unique Parameters

- **Search Children:** Triggers actions defined on the container that has the plugin attached to it on mouse events of child containers when set to On. When set to Off, only mouse events on the container that has the plugin attached to it trigger the actions.
- **Pick Button:** Defines which mouse button (Left, Middle or Right) triggers the Press and Release events.
- **On Press:** Defines the action that is triggered when the selected button is pressed and the cursor is within the container area.
- **On Release:** Defines the action that is triggered when the selected button is released and the cursor is within the container area.
- **On Enter:** Defines the action that is triggered when the cursor enters the container area.
- **On Leave:** Defines the action that is triggered when the cursor leaves the container area.
- **On Mouse Move:** To be implemented.
- **On Mouse Over:** Defines the action that is triggered while the cursor hovers within the container area.
- **On Mouse Drag:** Defines the action that is triggered while the container is dragged.
- **On Mouse Drop:** Defines the action that is triggered when another object is dropped within the container area.

- **Delay Click:** Delays actions triggered by the mouse by five fields when set to `On`. When set to `Off`, no delay is used.
- **Drag For Position:** Enables to container to be dragged and re-positioned in the render window when in On Air mode. Positioning occurs when the left mouse button is pressed and the mouse is dragged in the render window when the related container is selected. When set to `Yes`, additional parameters are enabled.
- **Use Drop Z-Buffer Value:** Changes perspective and container area when dragging an object around. This parameter defines whether the cursor location follows the container area through out the dragging of the object or the object moves according to the mouse movements. Due to perspective changes, the cursor might exit the container area while dragging.
- **Limit X Position:** Defines whether to limit the movement of the object on the X axis while dragging. When set to `Yes`, **Min X Position** and **Max X Position** are enabled, defining the minimum/maximum position on the X axis that the object can be dragged to.
- **Limit Y Position:** Defines whether to limit the movement of the object on the Y axis while dragging. When set to `Yes`, **Min Y Position** and **Max Y Position** are enabled.
- **Limit Z Position:** Defines whether to limit the movement of the object on the Y axis while dragging. When set to `Yes`, **Min Z Position** and **Max Z Position** are enabled.
- **Drag For Rotation:** Enables container to be dragged and rotated in the render window when in On Air mode. Rotation occurs when the center mouse button is pressed and the mouse is dragged in the render window when the related container is selected. When set to `Yes`, additional parameters are enabled.
- **Limit X Rotation:** Defines whether to limit the rotation of the object on the X axis while dragging. When set to `Yes`, **Min X Rotation** and **Max X Rotation** are enabled.
- **Limit Y Rotation:** Defines whether to limit the rotation of the object on the Y axis while dragging. When set to `Yes`, **Min Y Rotation** and **Max Y Rotation** are enabled.
- **Limit Z Rotation:** Defines whether to limit the rotation of the object on the Y axis while dragging. When set to `Yes`, **Min Z Rotation** and **Max Z Rotation** are enabled.
- **Rotation Sensitivity:** Defines the relation between the mouse movement and the object's rotation. The higher the sensitivity is, the larger mouse movements required to rotate the object to the same point.
- **Drag For Scale:** Enables to container to be scaled when the mouse is dragged in the render window when in On Air mode. Scaling occurs when the right mouse button is pressed and the mouse is dragged in the render window when the related container is selected. When set to `Yes`, additional parameters are enabled.
- **Limit Scale:** Defines whether to limit the scaling of the object while dragging. When set to `Yes`, **Min Scale** and **Max Scale** are enabled.
- **Scaling Sensitivity:** Defines the relation between the mouse movement and the object's scale. The higher the sensitivity, the larger mouse movements required to scale the object to the same size.
- **Use Another Container Bounding Box:** Enables Container For Bounding Box parameter when set to `Yes`. When set to `No`, the container's bounding box is used to capture mouse events.
- **Container For Bounding Box:** References this container's bounding box for events when a container is dragged to the placeholder.
- **Drop Type:** Defines which drop action triggers an event.

· **NONE:** Does not trigger an event.
· **Cont Geom:** Triggers a drag event for a dropped geometry.
· **Cont Image:** Triggers a drag event for a dropped image.
· **Other Cont Geom:** Enables a Dropped Container parameter when selected. When a container is dropped in the defined Dropped Container parameter, it triggers a drop event.
· **Other Cont Image:** Enables a Dropped Container parameter when selected. When an Image is dropped in the defined Dropped Container parameter, it triggers a drop event.
· **Use Viewports:** Enables a Camera Number parameter when set to `Yes`. Set the number of camera. The viewport area of the selected camera triggers the events.
· **Force Scene Redraw:** Redraws the render window every field when set to `Yes`. When set to `Off`, Viz renders the graphics normally.

6.2.22   DataImage

☐

DataImage changes the image or texture applied to the container.
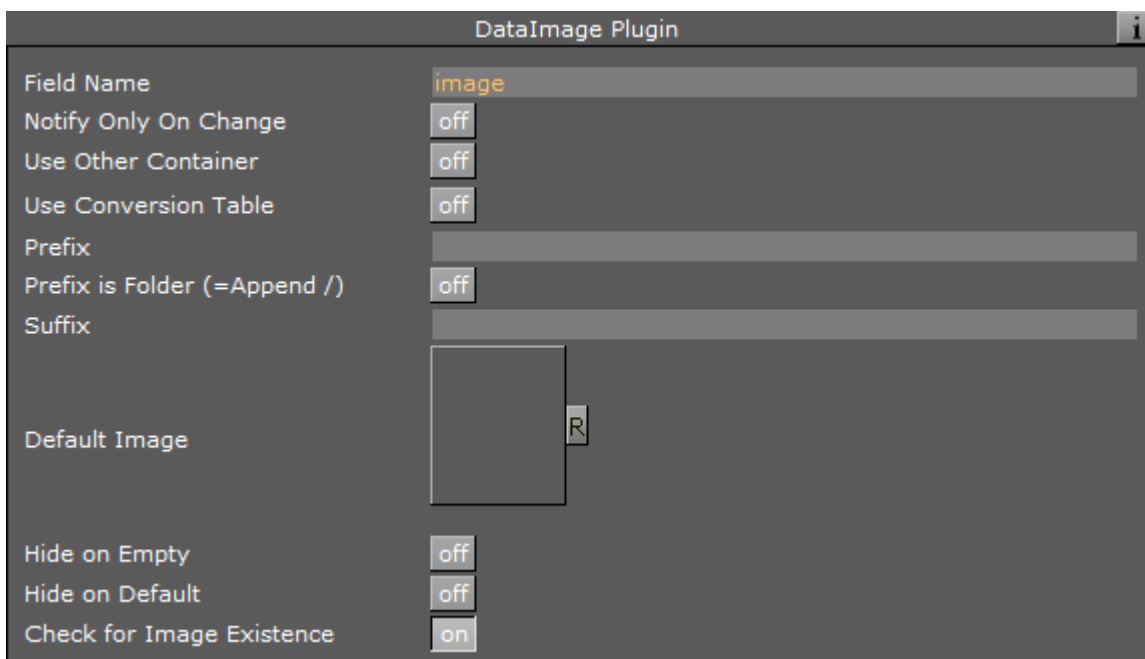
The data specifies the name of the image applied to the container.

Format

Image names should have the format:

· **IMAGE*<name>:** Where `<name>` is the full path to an image in Viz images library. The prefix `IMAGE*` can be omitted from the name.
· **BUILT_IN*IMAGE*<name>:** For special built-in images like VIDEO1, VIDEO2…

> ⚠ **Note:** The container must have an initial image attached to it for the plugin to take effect.

Example

A container with three children and with a DataImage plugin attached to it. If the Field Name is `IMAGES[3]` and the data is entered as `IMAGES[0-2]=IMAGE*image1, IMAGE*image2, BUILT_IN*IMAGE*VIDEO1`; then the first child shows image1, the second image2 and the third shows VIDEO1.

6.2.23   DataInRange

☐

DataInRange plugin converts a range of values into a single value. The value in the Field name parameter is compared to the defined values in the Range Descriptions parameter. If the input value is found in one of the ranges, it is converted to the value assigned to the range and the result is set in the Result Field Name.

> ⚠ **Note:** Range values must be numeric values.
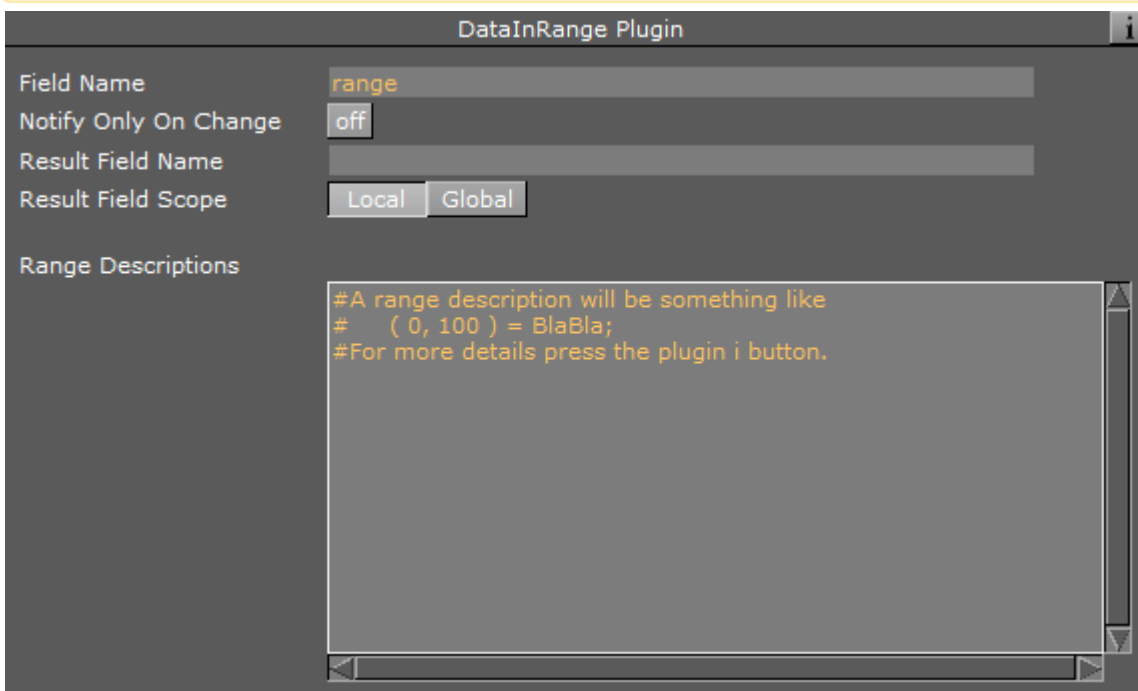
Unique Parameters

· **Result Field Name:** Defines the name of the DataPool variable that the result of the range conversion is assigned to.

- **Result Field Scope:** Defines the scope of result field. If set to `Local`, the Result Filed Name value is defined as part of a DataPool structure used in an object. If defined as `Global`, the Result field is defined as a regular data field.
- **Range Descriptions:** This parameter contains the different ranges. A range is defined inside brackets, with minimum and maximum values separated by comma. When a round bracket (opening or closing the range), the value next to it is not included in the range (but limiting the range). When square brackets are used the values are included in the range.

Example

In the following line the range of numbers between 1 and 10, excluding 1 and including 10, is converted to the text *Little Numbers*:

```
(1,10] = Little Numbers;
```

6.2.24 DataKey

DataKey toggles the Key function plugin.
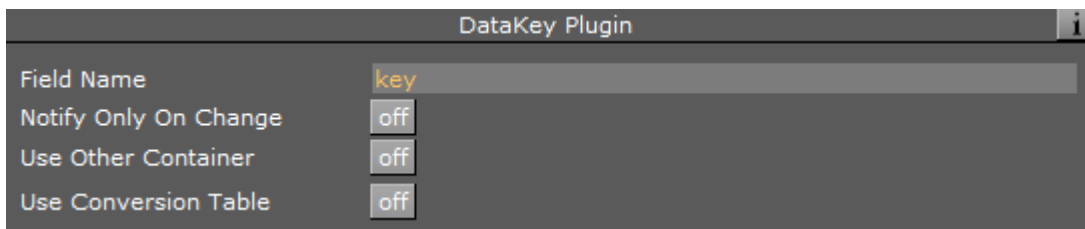
- **0:** Sets the key to `Off`.
- **Any other Value:** Sets the key to `On`.

> ⚠ **Note:** Key function plugin must be assigned to the container.

6.2.25 DataKeyFrame

DataKeyFrame affects a named keyframe of an animation defined on the container the plugin is attached to. Keyframe name must be unique. When the FieldName changes, the plugin looks for the keyframe and assigns new values to it.

Unique Parameters

- **KeyFrame Name:** Defines the name of the keyframe the plugin works on. The keyframe name must be unique.
- **Reset KeyFrame:** Defines the name of a keyframe containing initial parameters to be used when a RESET string is received. If a RESET data is accepted, the keyframe values are set to the values of the Reset KeyFrame. The keyframe name must be unique.
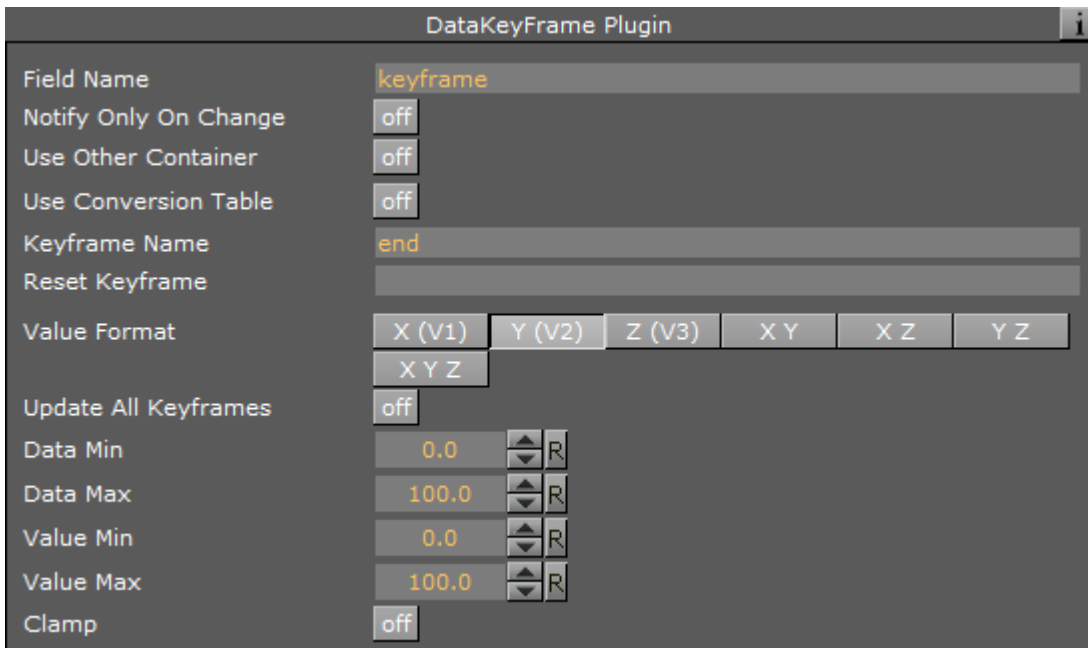- **Value Format:** Defines the format of the received data.

> ⚠ **Note:** If data format is not as defined in the Value Format parameter, unpredictable behavior occurs.

6.2.26   DataKeyFrame2

☐

DataKeyFrame2 behavior is the same as DataKeyFrame behavior, with the same parameters. The plugin was created because it is not possible to add the same function plugin twice to the same container.

> ⚠ **Note:** This plugin is deprecated. DataKeyFrame2 became obsolete when the parameter Use Remote Container was added to the DataPool plugins. The plugin is continued for backwards compatibility only, to support scene already using this plugin. When creating new scenes, use the DataKeyFrame plugin with the Use Remote Container option set to 0n.

6.2.27 DataKeyText

DataKeyText allows entering text to a container with text geometry using the keyboard. It allows defining actions to be taken as soon as the data arrives. In order for the data to arrive, the container must be *in focus*. The container is in focus when it is selected using the mouse.
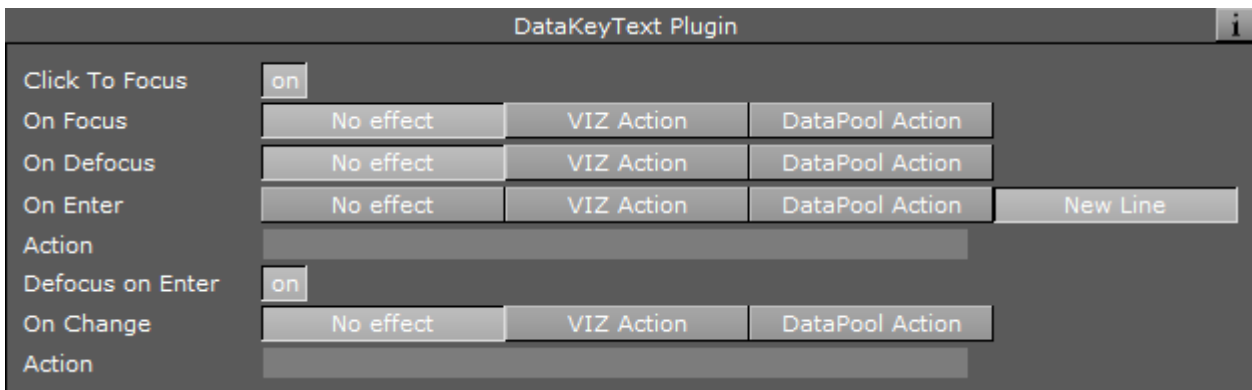
Unique Parameters

· **Click To Focus:** Specifies whether the container is always in focus or just when the mouse selects it.
· **On Focus:** Defines what to do when the container has focus. There are three options: First, nothing happens (no effect). Second, a Viz Action is invoked, or third, a DataPool Action is executed. DataPool Action is a series of commands in the same format as entered in the DataPool plugin.
· **On Defocus:** Defines what to do when the container loses focus. There are three options: First, nothing happens (no effect). Second, a Viz Action is invoked, or third, a DataPool Action is executed. DataPool Action is a series of commands in the same format as entered in the DataPool plugin.
· **On Enter:** Defines what to do when the container is in focus and the Carriage Return or **Enter** button is clicked. There are three options: First, nothing happens (no effect). Second, a Viz Action is invoked. Third, a DataPool Action is executed, or fourth, the new line is started in the text.
· **Defocus on Enter:** Removes container focus when a Carriage Return or **Enter** button is pressed and the On Defocus action is invoked.
· **On Change:** An action that is invoked every time a character is entered to the container. The options are No Effect, Viz Action or DataPool Action.

6.2.28 DataKeyTime

The DataKeyTime plugin moves a defined keyframe on the animation timeline to the time specified by the value of the DataField. The keyframe must have a unique name assigned to it. When used

with the Remote Container option, multiple keyframes of the same animation can be manipulated by multiple DataKeyTime plugins.
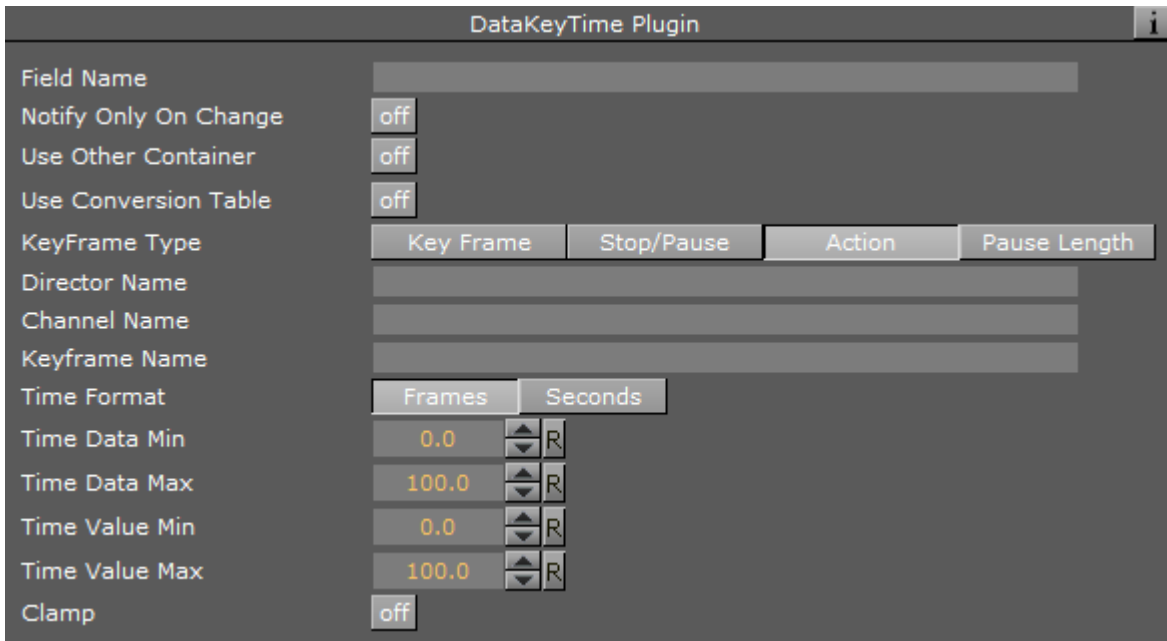
Unique Parameters

· **Keyframe Type:** Defines the type of animation property to affect: Keyframe, Stop or Pause point, or Action tag.

> ⚠ **Note:** Any property selected must be named uniquely. Otherwise the plugin does not affect the selected point.

· **Director Name:** Defines the name of the director containing the stop point or pause point to affect. The name is defined in the stage editor of Viz. This parameter is enabled when Key Frame Type is Stop/Pause Point or Action.
· **Channel Name:** Defines the name of the channel containing the action tag to affect. The name is defined in the stage editor of Viz. This parameter is enabled when Key Frame Type is Action.
· **Keyframe Name:** Defines the keyframe that is controlled by the plugin. The name is defined in the stage editor of Viz and it must be unique.
· **Time Format:** Defines if incoming data value is processed as Frame units (video frames) or seconds.

> ⚠ **Note:** Key Frames are moved to the absolute time value in the timeline. Abnormal behavior might occur when moving key frames on the timeline, since key frames can change their sequential order in the animation.

6.2.29 DataLookup

The DataLookup plugin searches data inside complex data objects using index and field name.

Unique Parameters

- **Source:** Defines the Data Object to be searching.
- **Attribute:** Defines the field name of Data Table inside Data Object after selected by index or hash key and get only value of this field.
- **Target:** Defines the DataField that receives the result of the operation.
- **Find Until:** Use when **Source** parameter assigned to DataHashTable with ordered numeric key and searching for element with maximum key value less than value of hash key
- **Return Key:** Use with **Find Until** parameter to get only the matched key value, not use **Attribute** parameter to get the field value inside.

6.2.30 DataLUImage

DataLUImage plugin defines a table of aliases which link to images in the Viz Images folders. The value sent in the DataField is one of the alias names. When the plugin receives the data, it compares the value to all the aliases defined in the table. If an alias matches the value sent, the plugin replaces the image on the controlled container to that image.
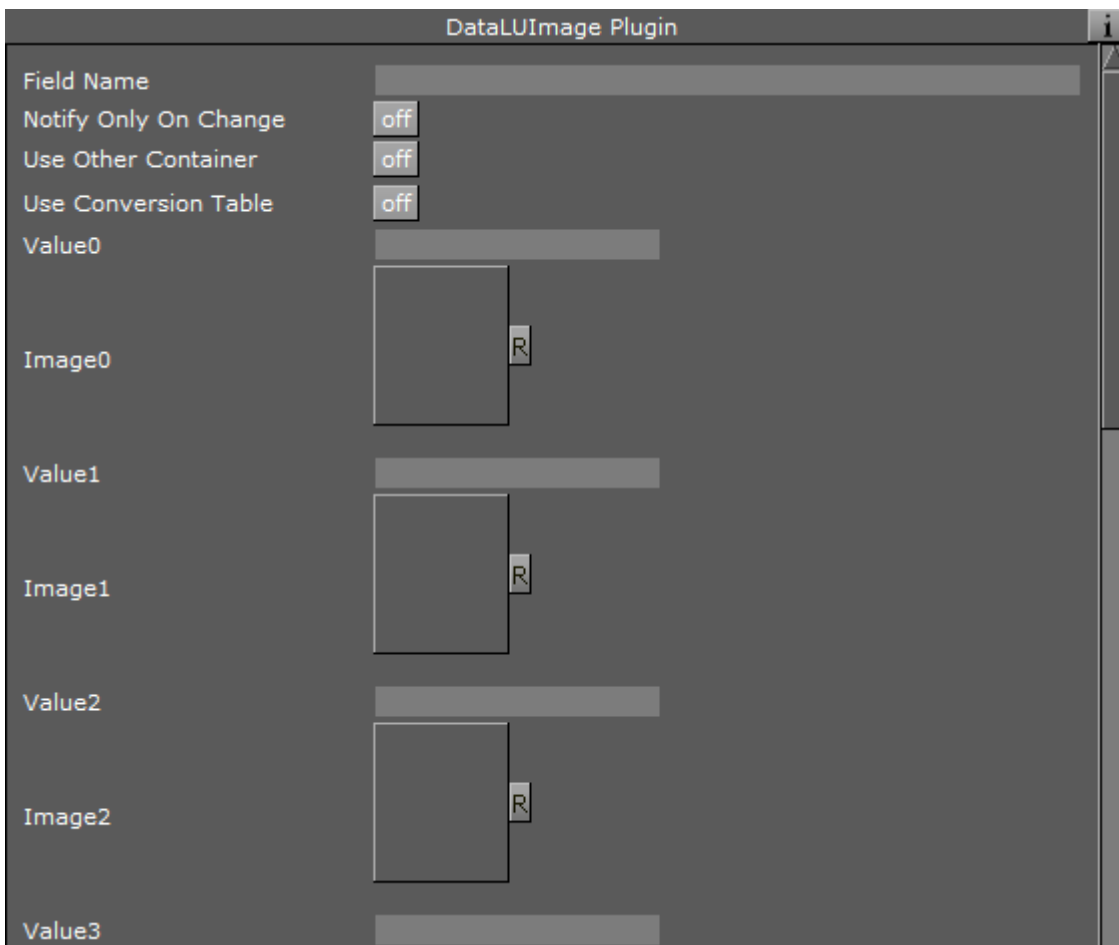
Unique Parameters

- **Image0..Image19:** Defines image placeholders for defining the images in the table. The images are dragged from the image library in Viz to the image place holders. To remove an image from the table, press the **Reset** button or drag a new image to it.
- **Value0..Value19:** Contains the aliases for each of the images.

Example

If Field Name is set to *MyImage*, then to change the image on the container, use the following command: `MyImage=Value1`; (Or any other value assigned to an image).

6.2.31   DataManipulate

☐

DataManipulate enables interactive manipulation of the object that the plugin is placed on, when in On Air mode. The left mouse button controls position manipulation, center Mouse button affects rotation and the right mouse button affects scaling.
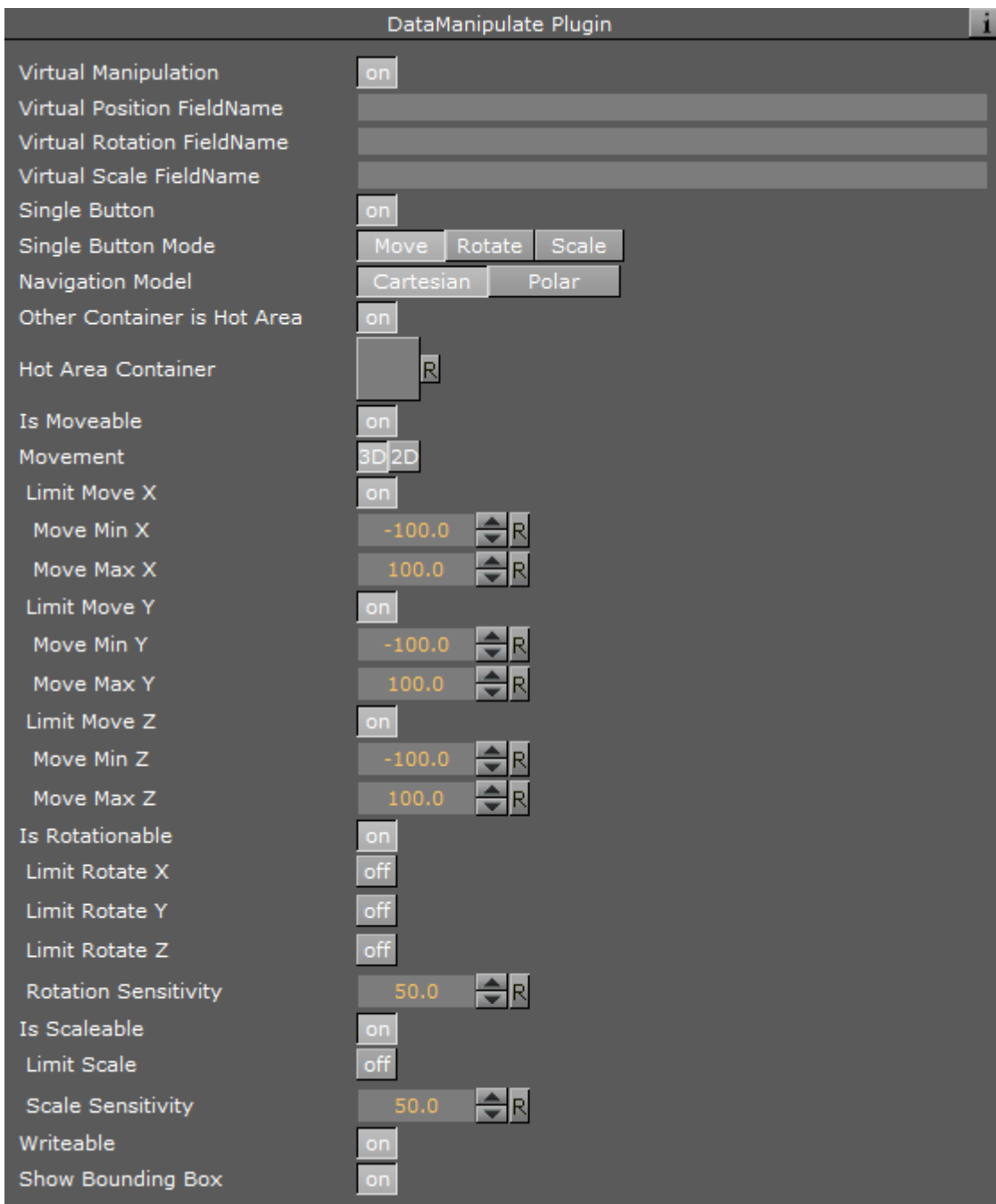
Unique Parameters

- **Virtual Manipulation:** Defines if object manipulation is based on Data Fields values. When set to `0n`, additional parameters are enabled:
  - **Virtual Position FieldName:** Defines a DataPool variable that controls the position of the manipulated container. The value of the variable defines the delta of the object position from the current position.

- **Virtual Rotation FieldName:** Defines a DataPool variable that controls the rotation of the manipulated container. The value of the variable defines the delta of the object rotation from the current rotation value.
- **Virtual Scale FieldName:** Defines a DataPool variable that controls the scale of the manipulated container. The value of the variable defines the delta of the object scale from the current scale.
- **Single Button:** Allows all mouse buttons to control one property of the object when set to On. An additional parameter is enabled, setting the controlled attribute of the object:
  - **Single Button Mode:** Selects the control mode of the mouse.
- **Navigation Model:** Selects the type of axis used to manipulate the object.
  - **Cartesian:** Uses the Cartesian coordinate system to calculate object movement.
  - **Polar:** Uses the Polar coordinate system to calculate object movement.
  - **Navigation Model:** Selects the type of axis used to manipulate the object.
- **Other Container is Hot Area:** Defines a remote container as the hot area instead of the container the plugin is attached to. When set to On, an additional parameter is enabled: Hot Area Container, and a container placeholder is added.
- **Is Moveable:** Moves the object when selected and dragged while the left mouse button is pressed, when set to On. When set Off the object does not move when dragged. Additional parameters are enabled when set to On:
  - **Movement:** Determines how the selected object can move. When 2D is selected, the object can move only on the X and Y axes. When 3D is selected, the object can be moved in all three axes.
  - **Limit Move X:** Limits the object's position values according to the values set in the Move Min X and Move Max X when set to On. When set to Off the object can be dragged all over the render window.
  - **Limit Move Y:** Limits the object's position values according to the values set in the Move Min Y and Move Max Y when set to On. When set to Off the object can be dragged all over the render window.
  - **Limit Move Z:** Limits the object's position values according to the values set in the Move Min Z and Move Max Z when set to On. When set to Off the object can be dragged all over the render window.
- **Is Rotation able:** Rotates the object when selected and dragged while the center mouse button is pressed, when set to On. When set Off the object does not rotate when dragged. Additional parameters are enabled when set to On:
  - **Limit Rotate X:** Limits the object's rotation values according to the values set in the Rotate Min X and Rotate Max X when set to On. When set to Off the object can be rotated to any angle.
  - **Limit Rotate Y:** Limits the object's rotation values according to the values set in the Rotate Min Y and Rotate Max Y when set to On. When set to Off the object can be rotated to any angle.
  - **Limit Rotate Z:** Limits the object's rotation values according to the values set in the Rotate Min Z and Rotate Max Z when set to On. When set to Off the object can be rotated to any angle.
  - **Rotation Sensitivity:** Defines the relation between the mouse movement size and the rotation size.

- **Is Scaleable:** Scales the object when selected and dragged while the right mouse button is pressed, when set to `On`. When set `Off` the object does not scale when dragged. Additional parameters are enabled when set to `On`:
    - **Limit Scale:** Limits the object's scaling values according to the values set in the Min Scale and Max Scale, when set to `On`. When set to `Off` the object can be scaled to any size.
    - **Scale Sensitivity:** Defines the relation between the mouse movement size and the scaling size.
- **Writeable:** Controls a text object. If the plugin is controlling a text object and writable is `On`, when selecting the object the user can type and change the content of the text value. When set to `Off`, the text object cannot be modified.
- **Show Bounding Box:** Toggles visibility for the object's bounding box. When set to `On`, the object's bounding box is visible when the object is selected (in On Air mode). When set to `Off`, the bounding box is not visible when the object is selected.

> ⚠ **Note:** DataManipulate plugin does not work if the scene has a DataInteractive plugin added.

6.2.32 DataMaterial

DataMaterial changes the material parameters of the controlled container. The data specifies any combination of the ambient, diffuse, specular, emission and alpha parameters of the material. The data format is:

```
[AMBIENT r g b] [DIFFUSE r g b] [SPECULAR r g b] [EMISSION r g b] [ALPHA a]
```

This format can specify new values for separate material components.

- **[COLOR r g b]:** Specifies a color using red green and blue values.

> ⚠ **Note:** All color values (r g b) are numbers between `0` and `255`. Alpha values are float numbers between `0` and `100`.

- **[MATERIAL name]:** Specifies a material name from the Viz material library. The message format is: `MATERIAL MATERIAL*<name>`, where `<name>` is the full path to a material in Viz

material library.
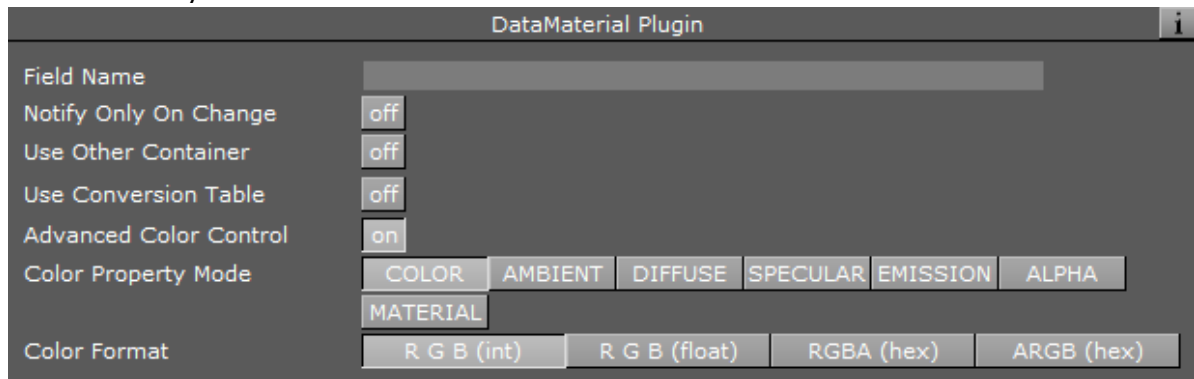
Example

- If the received data is `MAT[0-3]=DIFFUSE 255 0 0, DIFFUSE 0 255 0, DIFFUSE 0 0 255;` then the diffuse component of the material of the first child is red, the diffuse component of the material of the second child is green and, the diffuse component of the material of the third child is blue.
- If the data received is `MAT=COLOR 255 0 0;` then all the container materials are set to red.
- If the data received is `MATERIAL MATERIAL*blue;` then the material is set to the blue material in Viz material library.

> ⚠ **Note:** DataMaterial plugin does not work unless the controlled containers have a material on them. If controlling child containers from a top node, each of the controlled child containers must have a material attached to it. The top container, hosting the DataMaterial plugin, doesn't need to have a material attached to it.

6.2.33 DataMaterialGradient

DataMaterialGradiant changes the material color of the controlled container to a color within a predefined spectrum (the plugin changes the hue of the material). A minimum value is assigned to one color and a maximum value is assigned to another color. Any value between the minimum and maximum values is translated to a color, and affects the container's material.
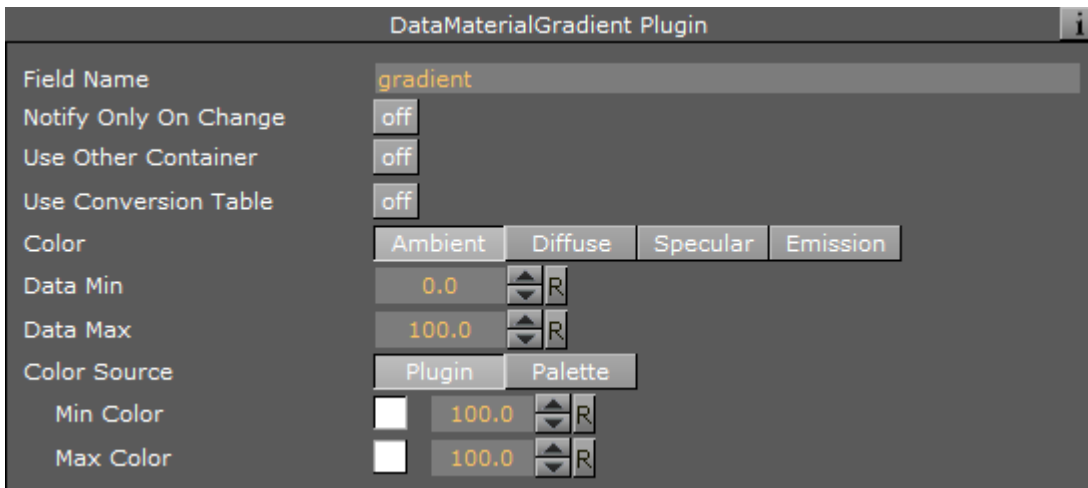
Unique Parameters

- · **Source material:** Assigns the Data Min value to the material/alpha.
- · **Target material:** Assigns the Data Max value to the material/alpha.

Example

Data Min is 0 and Data Max is 100. Source material is red and Target material is green. If the received data is 50 the material that is set on the container is yellow.

> ⚠ **Note:** DataMaterialGradiant plugin does not work unless the controlled containers have a material on them. If controlling child containers from a top node, each of the controlled child containers must have a material attached to it. The top container hosting the DataMaterialGradiant plugin doesn't need to have a material attached to it.

6.2.34   DataMaterialIndex

DataMaterialIndex plugin changes the material of the controlled container. The scene must have a DataMaterialTable scene plugin with a defined material table. The DataMaterialIndex plugin uses the indexes of the defined materials in the DataMaterialTable to change the material on the containers. Data format is an index number (integer). The DataMaterialIndex plugin copies the material related to the given index from the table to the controlled container. The controlled container must have material attached to it.
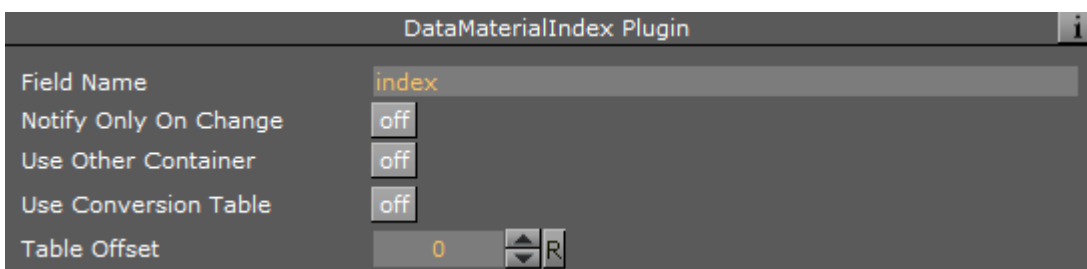
Example

```
MatIndex[0-9]=1,2,3,4,5,6,7;
```

The material on the first child is changed to the material defined in entry 1 in the material table, the second is changed to the material defined in entry 2 of the material table, and so on. The last containers (index 7-9) are set to the material defined in index 7 of the table.
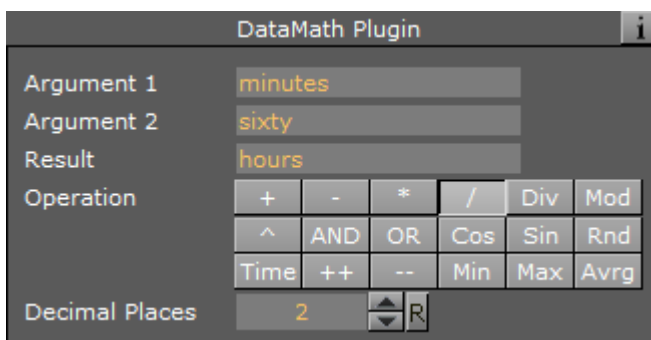
6.2.35   DataMath

DataMath plugin performs mathematical calculations with DataPool variables and writes the result to a specified DataField.

Unique Parameters

- **Argument 1 and Argument 2:** Defines the arguments used for the mathematical operation. *argument1* is used at the left hand side of the operator. Both arguments must be DataPool variables (DataFields) or constants. If the operation is unitary, only *argument1* is used.

  > ⚠ **Note:** Both arguments (DataFields) trigger the DataMath plugin and the result is calculated.

- **Result:** Defines the DataField that receives the result of the operation.
- **Operation:** Defines the operator that is used for the calculation.
    - **+ - *:** Adds/subtracts/multiplies the arguments.
    - **/:** Divides *argument1* by *argument2*.
    - **Div:** Provides integral part of the division of *argument1* by *argument2*.
    - **Mod:** Gives the remainder of the division of *argument1* by *argument2*.
    - **^:** Raises *argument1* to the power of *argument2*.
    - **AND:** Performs a logical AND of the arguments: If both are different than zero the result is `1`.
    - **OR:** Performs a logical OR of the arguments: If both are zero the result is `0`.
    - **Cos / Sin:** Determines cosines/sines of *argument1*.
    - **Rnd:** Gives a random value between the two values.

- **Time ++/--:** Increases/decreases *argument1* by one.
- **Min/Max:** Performs a numeric comparison of the arguments and returns the lower/higher value.
- **Avrg:** Calculates the average of the arguments.
- **Decimal Places:** Determines the number of decimal places for the result of the operation.

6.2.36   DataMathObject

DataMathObject plugin performs mathematical calculations with DataPool objects and variables and writes the result to a specified Result.
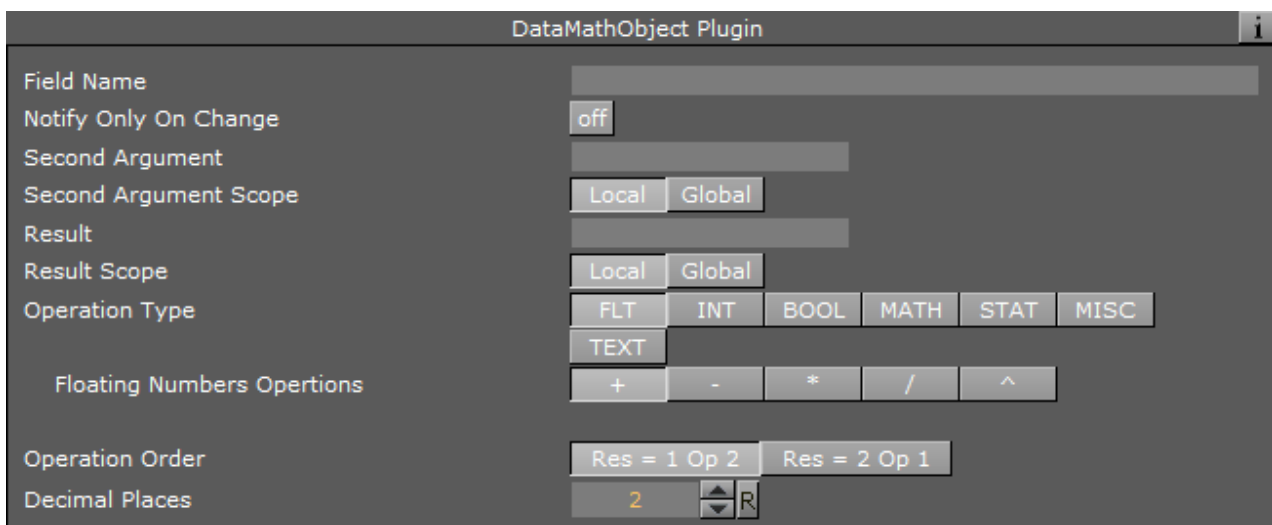
Unique Parameters

- **Field Name:** Defines the first argument for the mathematical operation. This argument is a DataPool variable or a DataPool Object.
- **Second Argument:** Defines the second argument used for the mathematical operation. The order of the arguments in the mathematical expression is defined in the Operation Order argument.
- **Second Argument Scope:** Defines the scope of the second argument, i.e. whether the argument is a local variable (part of an object) or a global DataPool variable.
- **Result:** Defines the DataField that receives the result of the mathematical operation.
- **Result Scope:** Defines the scope of the result, i.e. whether the argument is a local variable (part of an object) or a global DataPool variable.
- **Operation Type:** Defines type of operation to perform. According to the selected operation type, a list of operations are displayed. The selected operator is used in the mathematical expression.
- **Operation Order:** Defines the order of the arguments in the mathematical expression.
- **Decimal Places:** Determines the number of decimal places for the result of the operation.

Notes

- If the operation is unitary, only *argument1* is used.
- Only the Field Name parameter triggers the DataMathObject plugin and the result is calculated.
- If one of the variables is not defined it is ignored (it is not created automatically like in other DataPool plugins).

6.2.37 DataMinMax

The DataMinMax plugin receives a DataPool array name and returns two DataFields: One containing the minimum value of the array and the other containing the maximum value of the array.
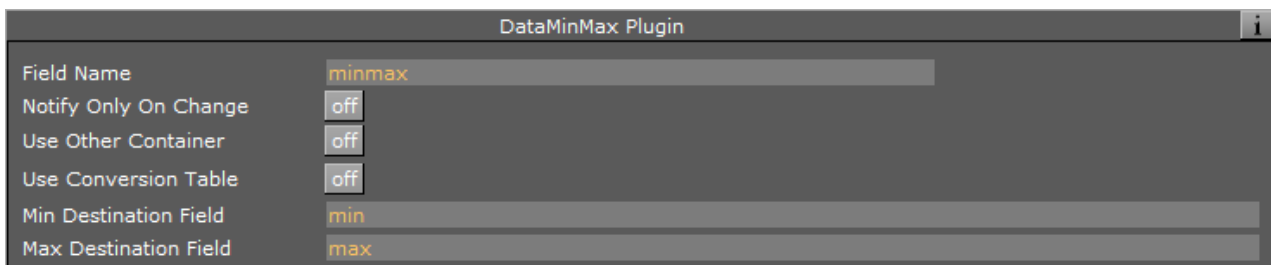
Unique Parameters

- **Field Name:** Defines the name of the array to be checked.
- **Min Destination Field:** Assigns the minimum value of the array into a DataField.
- **Max Destination Field:** Assigns the maximum value of the array into a DataField.

6.2.38 DataMouseAction

The DataMouseAction plugin receives mouse events from the DataMouseSensor plugin and runs Viz actions accordingly. This plugin is not registered to a DataField. It is attached to a container and it is triggered by the mouse actions when the cursor is within the container area in the render window.

> ⚠ **Note:** DataPool commands cannot be used.
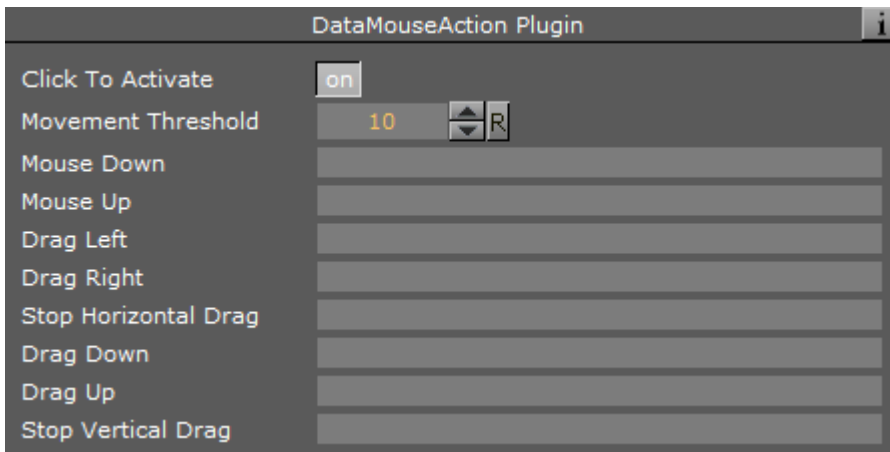
Unique Parameters

- **Click To Activate:** Determines whether the user has to select the container in order to activate the actions. If set to 0n, it is selected when the first action that is executed (the Mouse Down action).
- **Mouse Threshold:** Determines the minimum number of pixels that the mouse arrow has to be moved before a movement is recognized.
- **Mouse Down:** Calls a Viz action as soon as a mouse button is pressed on top of the container the plugin is attached to.
- **Mouse Up:** Calls a Viz action when a mouse button is released and the container the plugin is attached to is the previously selected container.
- **Drag Left:** Calls a Viz action when the selected container is dragged (moved) to the left and the movement is larger then the Mouse Threshold value.
- **Drag Right:** Calls a Viz action when the selected container is dragged (moved) to the right and the movement is larger then the Mouse Threshold value.
- **Stop Horizontal Drag:** Calls a Viz action when a horizontal mouse dragging movement stops or the mouse button that started all the drag movement stops.
- **Drag Down:** Calls a Viz action when the selected container is dragged (moved) down and the movement is larger then the Mouse Threshold value.
- **Drag Up:** Calls a Viz action when the selected container is dragged (moved) up and the movement is larger then the Mouse Threshold value.
- **Stop Vertical Drag:** Calls a Viz action when a vertical mouse dragging movement stops or the mouse button that started all the drag movements stop.

6.2.39 DataMousePosition

The DataMousePosition plugin receives mouse events from the DataMouseSensor plugin and enables dragging the container to attach it to, when in On Air mode. This plugin is not registered to a DataField. It's attached to a container and it is triggered by the mouse actions when the container is in focus (selected).
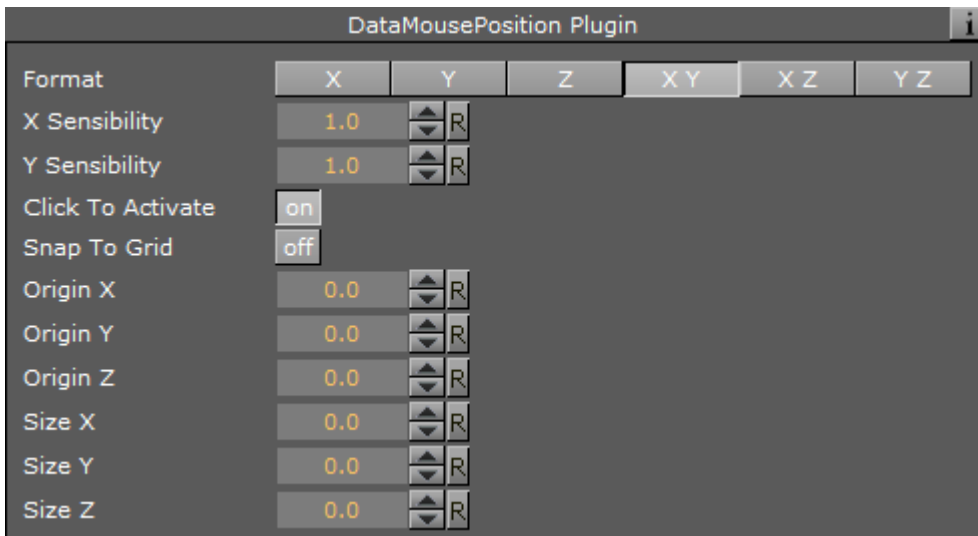
Unique Parameters

- **Format:** Specifies in which axis or plane the container moves.
- **X Sensibility:** Defines the sensibility of the mouse movement in the X-axis.
- **Y Sensibility:** Defines the sensibility of the mouse movement in the Y-axis.
- **Click To Activate:** Defines whether the user must click on the container in order to activate it.
- **Snap To Grid:** Specifies whether the object snaps to an imaginary grid whenever the movement is over.
- **Origin X, Y and Z:** Defines the origin of the imaginary grid to be used when Snap To Grid is on.
- **Size X, Y and Z:** Defines the dimensions of the cells of the imaginary grid to be used when Snap To Grid is on.

6.2.40   DataMultiParam

DataMultiParam plugin extends the DataParameter plugin options. It enables DataPool to control multiple parameters. Incoming data contains all the values for the defined parameters.
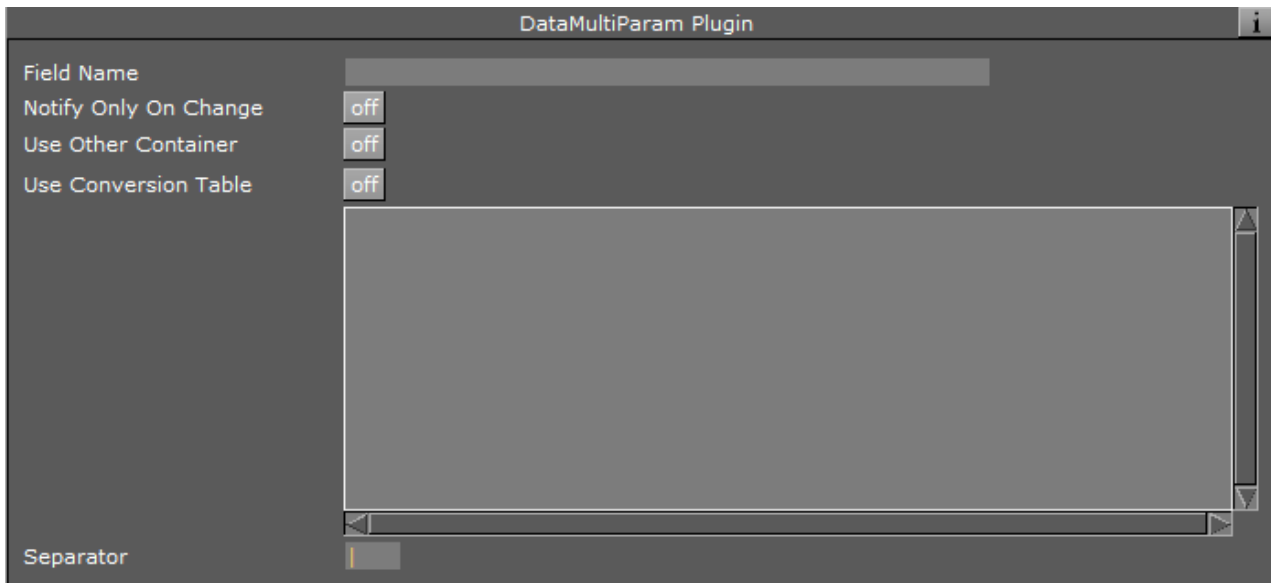
Unique Parameters

- **Format:** Defines the container parameters to be controlled by incoming data. The format is specified in lines, each containing the following information:

  `<Plugin Type> <Plugin Name> <Parameter Name>`

  - `<Plugin Type>` is either GEOMETRY or FUNCTION.
  - `<Plugin Name>` is the name of the plugin that is controlled.
  - `<Parameter Name>` is the name of the parameter in the specified plugin to control.

  > ⚠ **Note:** Plugin name and parameter name must be identical to the names used in Viz (names are case sensitive). To find out the correct names, open the Viz command console and change the parameter required. The correct spelling is displayed in the command console as part of the messages sent from Viz.

- **Separator:** Defines a character in the incoming data that separates between the values of the defined parameters. Default separator is a pipe character (|).

Example

If the DataMultiParam plugin is attached to a cylinder object, with an AutoRotate function plugin, Field name is *VALUE* and the controlled parameters are:

- GEOMETRY Cylinder angle
- GEOMETRY Cylinder hole
- FUNCTION AutoRotate rotspeed

Using the default separator, incoming data is `VALUE=90 | 100 | 230`. The result is:

- Parameter *angle* of the Cylinder is set to the value `90`.
- Parameter *hole* of the Cylinder is set to the value `100`.
- Parameter *rotspeed* of the AutoRotate plugin is set to the value `230`.

6.2.41   DataNumber

DataNumber works on containers with text geometry. The plugin formats the display of numeric values according to the parameters set in the plugin.
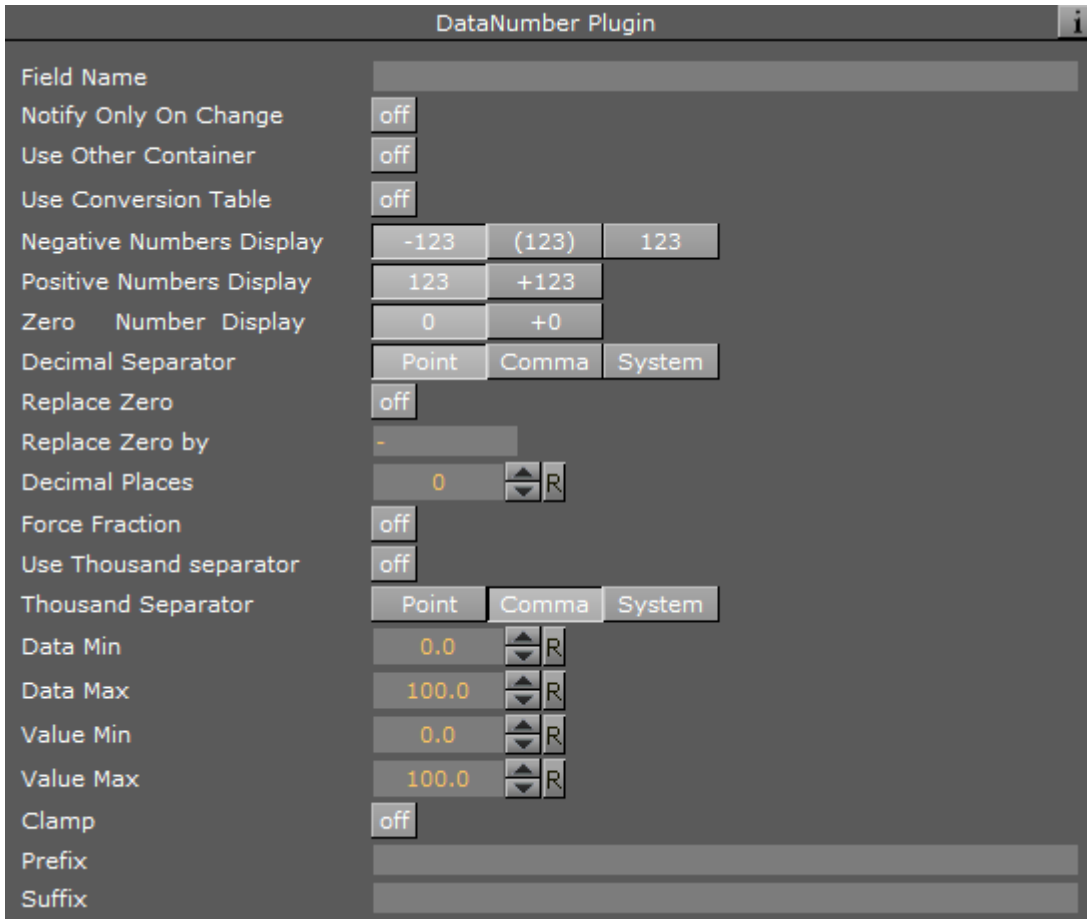
Unique Parameters

- **Negative Numbers:** Defines whether to show a negative number with a minus prefix -123,0 or use the parenthesis format (123).
- **Decimal Separator:** Defines which character to use as a decimal separator: a comma or a dot.
- **Replace Zero:** Defines whether to replace a zero value with the string defined in the Replace Zero By parameter.
- **Replace Zero By:** Defines a string that is displayed when the numeric value equals zero.
- **Decimal Places:** Defines how many digits are displayed on the right side of the decimal separator.
- **Force Fraction:** Shows a fraction with the defined decimal places even though the value is an integer or has a fraction with less digits than defined when enabled.
- **Use Thousand Separator:** Defines whether a thousand separator is displayed.

- **Thousand Separator:** Defines which character to use as a thousand separator: A comma or a dot.

> ⚠ **Note:** When displaying accounting information it is common to use the parenthesis to show negative numbers.

> ⚠ **Note:** The fraction is rounded according to the value defined in the Decimal places parameter.

6.2.42 DataObject

☐

DataObject allows to create a new typed object and link it to a Viz hierarchy. When a DataObject plugin is attached to a container, a field name and an object type is specified the DataObject plugin searches Viz hierarchy to look for a parent for the object and children underneath. Every Data plugin found under the container where the DataObject is found is tested to see whether it's a field of the object. If it is so, it is attached as a child of the object.
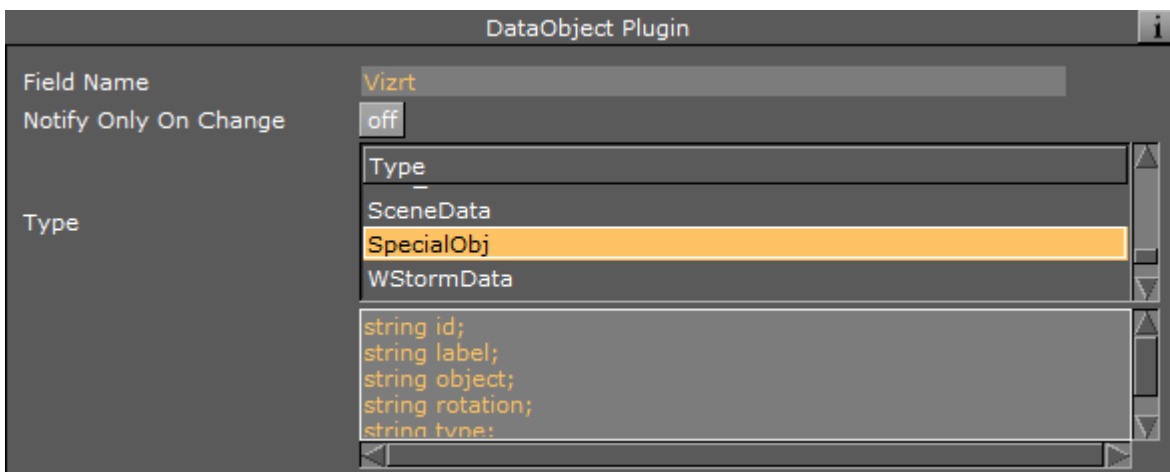
Unique Parameters

- **Type:** Defines the type of the object. The list of the types is taken from the configuration file.

6.2.43 DataObjectTracker

☐

DataObjectTracker plugin tracks the controlled container and sends the selected attribute values to a defined DataPool variable (DataField).

Unique Parameters

- **Destination Field:** Assigns container values to *DataObjectTracker* from the named *DataField*.
- **Track Type:** Defines which container attributes are tracked by the *DataObjectTracker*.
- **Format:** Defines the format by which the attribute values are sent to the Destination Field.
- **Track when hidden:** Defines if the object tracks when it is hidden.
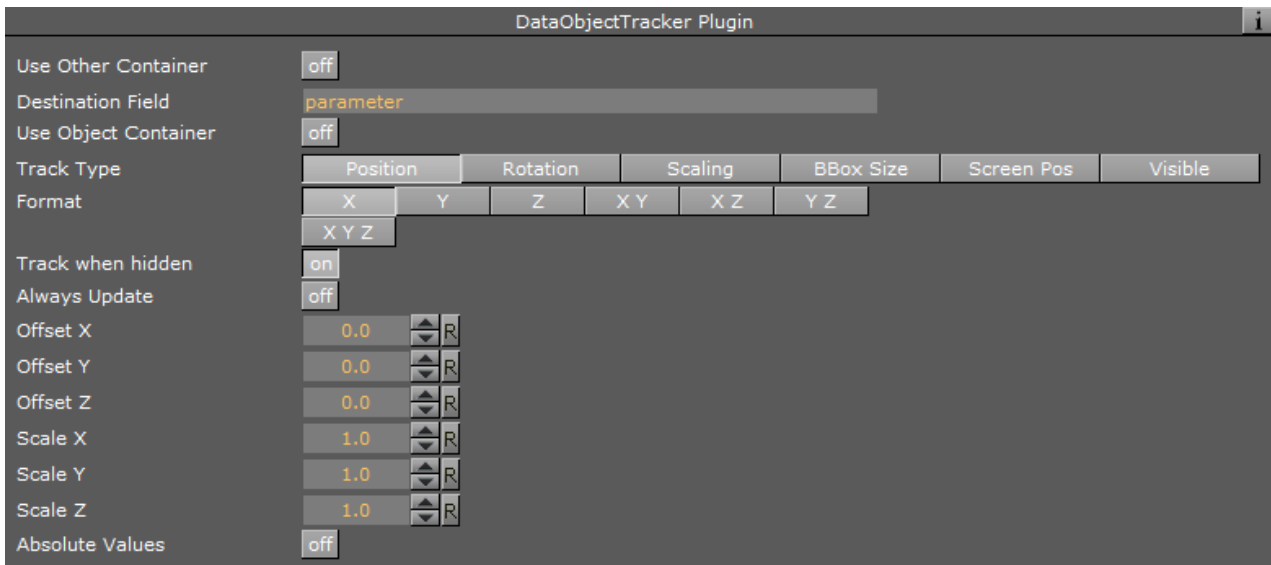- **Offset X/Y/Z:** Adds a fixed value to the current values of the container, in the X/Y/Z axis, before setting the values in the Destination Field.
- **Scale X/Y/Z:** Multiplies a fixed value by the current values of the container's scale, in the X/Y/Z axis, before setting the values in the Destination Field.
- **Absolute Position:** Refers all read values from the container in relation to its top parent node (i.e. the returned values include all position, rotation or scaling values of all parent containers of the controlled container) when set to `On`. When set to `Off`, all values read from the container refer to the container itself in relation to its position, rotation or scaling.

6.2.44   DataParameter

☐

DataParameter plugin controls other plugins (geometry or function plugins) parameter values.

Unique Parameters

- **Parameter Type:** Defines the controlled plugin type. The options are GEOMETRY, FUNCTION or SCENE. If Function or Scene is selected, the Function Name parameter is enabled.
- **Function Name:** Sets the name of the function (or scene) plugin that the DataParameter plugin controls.
- **Parameter Name:** Defines the parameter that is controlled. The parameter name must be identical to the name used by Viz (case sensitive). To find the exact parameter name use the show command option and change the parameter from the user interface. The name of the parameter is printed by Viz in the commands console.

Example

Suppose a group has three children, each child is a cube. If we want to change the height parameter of each cube according to a certain value received from the field VALUES[3]. Field Name = VALUES[3], Parameter Type = GEOMETRY (cube is a geometry plugin), Function Name = Cube and Parameter Name=size_Y (name of the parameter in the cube plugin that changes the height of the cube) If the data is VALUES[0-2]=1, 2, 3; the cube's heights becomes 1, 2 and 3 respectively.

6.2.45 DataParamTracker

☐

DataParamTracker plugin tracks the controlled container and sends the defined parameter values to a defined DataPool variable (DataField).
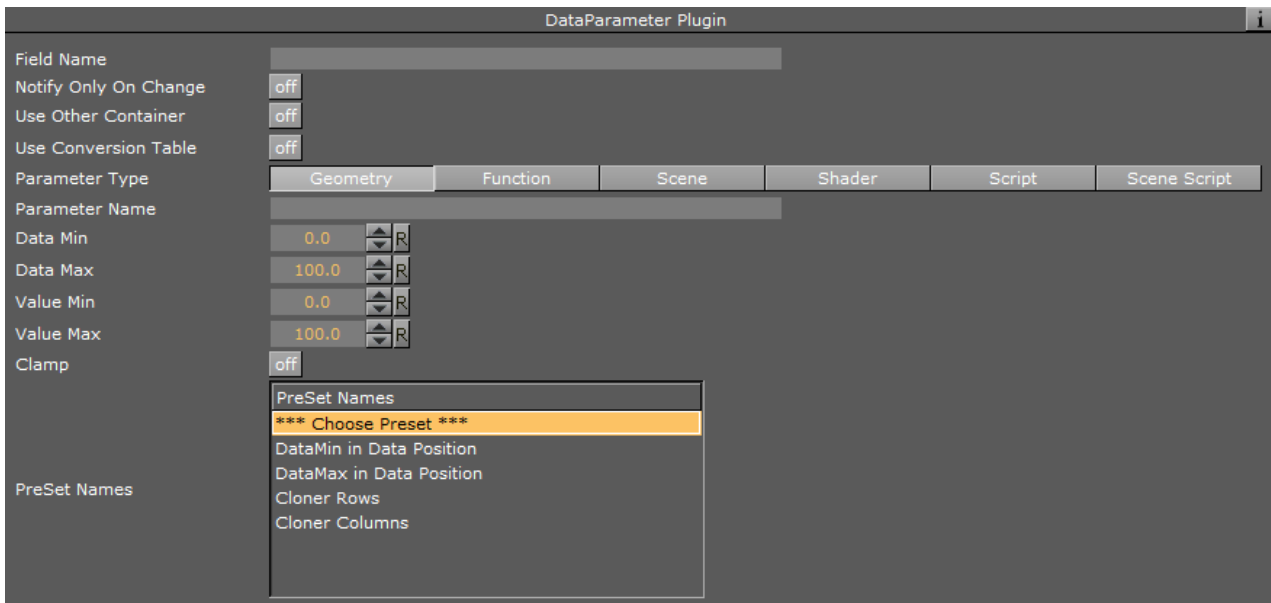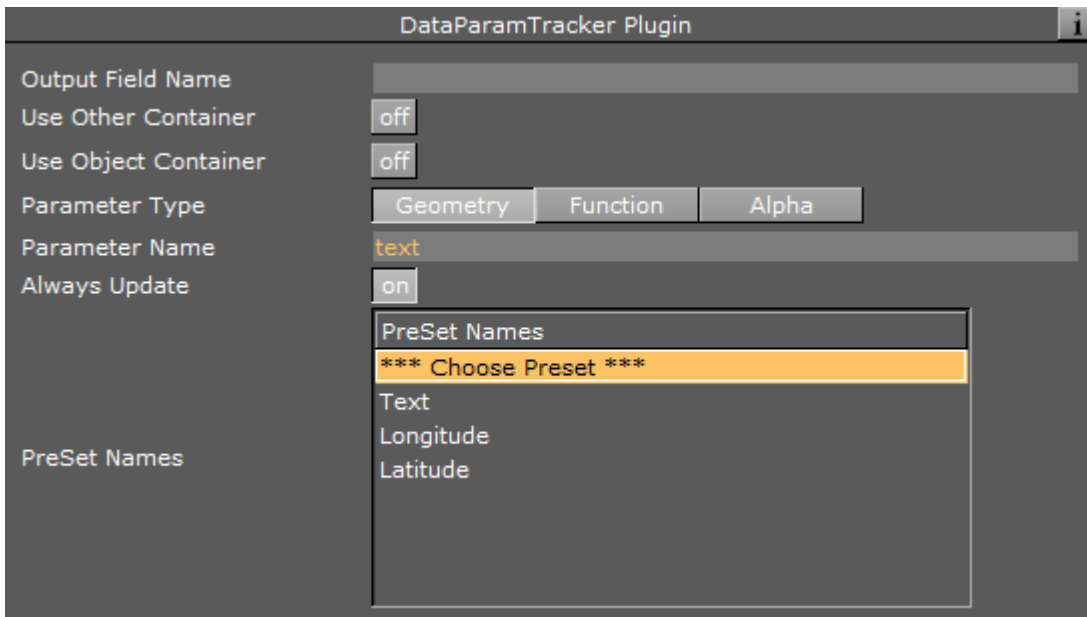
Unique Parameters

- **Destination Field:** Assigns container parameter values to *DataParamTracker* according to the name of the *DataField*.
- **Parameter Type:** Defines the type of the tracked plugin.
- **Function Name:** Defines what function plugin is tracked. The function plugin name must be identical to the name used by Viz (case sensitive). To find the exact parameter name use the show commands console.
- **Parameter Name:** Defines the parameter that is tracked. The parameter name must be identical to the name used by Viz (case sensitive). To find the exact parameter name use the SHOW COMMAND option and change the parameter from the user interface. The name of the parameter is printed by Viz in the command console.

6.2.46 DataPosition

DataPosition changes the position of the container according to the received data.
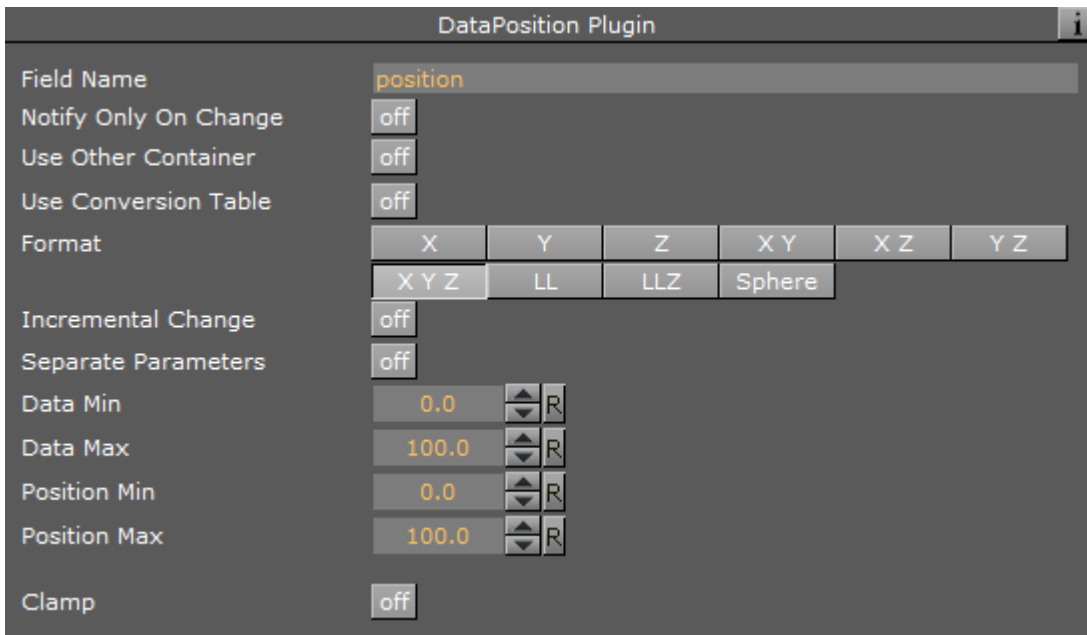
Unique Parameters

- **Format:** Specifies the format of the incoming data. Format options specify the axis which the data relates to.
- **Incremental change:** Defines if the received data is added to the current value of the data field. When set to On the received data is added to the data field. When set to Off the received data replaces the value of the data field.
- **Separate Parameters:** Enables additional parameters when set to On, allowing the user to define minimum and maximum values to all parameters separately:
  - **X Min/Max:** Defines a minimum/maximum value for the received X position of the object.
  - **X Position Min/Max:** Defines a minimum/maximum Viz value for the received X position of the object.

> ⚠ **Note:** (Y and Z axis have the same parameters. The parameters are enabled according to the selected format)

Example

If Field Name is VALUE and the format is X, the data sent is VALUE=10.5;. The container moves on the X axis to a position of 10.5.

6.2.47 DataReader

DataReader is a plugin that allows for reading feeds of information from several different types of sources of information. The plugin supports reading Excel files, databases using SQL, feeds using

XML and JSON. The plugin also supports different types of outputs. The plugin shows different arguments depending on the inputs and the outputs used:

- Functionality
- Input Type - Excel
- Example
- Input Type - SQL
- Input Type - XML
- Input Type - JSON

Functionality

All of the different sources of information behave the same way: They all search for records and translate the records fields to a certain format according to the requested output type. In the case of DataPool output types, the plugin converts each incoming record to a DataPool command and executes it. In the case of Shared Memory, the plugin creates a character separated string of fields and pushes to shared memory.

Input Type – Excel

In this mode the plugin reads a sheet from an Excel file. The parameters in this mode are:

- **File Name:** Determines the name of the file to be read.
- **Table/Sheet:** Determines the name of the sheet to be read from within the file.
- **Key:** Determines the name of the column in the Excel file from which the keys to the DataTable should be taken.
- **From row:** Determines the row number in the Excel sheet from where to start reading the records.
- **Number of rows:** Determines the maximum number of records to read, starting from the row stated in *From row*.
- **Fields to Read:** Specifies by name the columns to read.
- **DP Field Name:** Specifies the field name in DataPool where to write the output.
- **Field is a:** Specifies the type of the target.

- **Post Update Action:** Specifies commands (either DataPool or Viz commands) to be executed right after the data is read and sent to the target.
- **Show Data:** Dumps the data the plugin reads to a DataPool variable called `READER_CONSOLE` when set to `On`. This is extremely useful for development and debugging purposes. This dump includes error messages too.
- **Return Variables Prefix:** Specifies a prefix to add to the name of the DataPool variables that the plugin uses. For example, if the prefix is `STV_`, then the plugin dumps the debug information to `STV_READER_CONSOLE`.
- **Load Automatically:** Reads the information automatically every certain amount of time. This amount of time is defined in *Automatic Load Period (in seconds)*.
- **Automatic Load Period (in seconds):** Specifies the period of time between consecutive automatic reads.
- **Load:** Invokes the action of reading the data.

Example

The following is a typical example of a table in an Excel sheet we would be interested in reading from using *DataReader*.

In this case, if the parameters are configured as:

- File name = C:\temp\Indexes.xlsx
- Table/Sheet = Indexes$
- Key = Stocks
- From Row = 5
- Number of Rows = 3
- Fields To Read = Stocks, Prices
- DP Field Name = Indexes
- Data Field = DataTable
  then the plugin dumps to DataPool the following command:

```
Indexes[EURO STOXX 50 ]={
        {
                Stocks=""EURO STOXX 50 "";
                Prices=""3131.39"";
                Volumes=""-37.84 (-1.19%)"";
        };
};

Indexes[CAC 40 ]={
        {
                Stocks=""CAC 40 "";
                Prices=""4265.04"";
                Volumes=""-47.26 (-1.10%)"";
        };
};

Indexes[S&P TSX ]={
        {
                Stocks=""S&P TSX "";
                Prices=""15476.77"";
                Volumes=""-48.05 (-0.31%)"";
        };
};

NUM_RECORDS=3;
```

Note that the *Key* parameter is used to obtain the column from which the indexes of the table.

· In case the Data Field is of type **DataArray:**

```
Indexes[0]={
        {
                Stocks=""EURO STOXX 50 "";
                Prices=""3131.39"";
                Volumes=""-37.84 (-1.19%)"";
        };
};

Indexes[1]={
        {
                Stocks=""CAC 40 "";
                Prices=""4265.04"";
                Volumes=""-47.26 (-1.10%)"";
        };
};

Indexes[2]={
        {
                Stocks=""S&P TSX "";
                Prices=""15476.77"";
                Volumes=""-48.05 (-0.31%)"";
        };
};

NUM_RECORDS=3;
```

- In case the Data Field is of type **DataStructure**, the *Key* parameter is used as the name of the column from which to take the names of the DataPool field:

```
EURO STOXX 50={
        Stocks=""EURO STOXX 50 "";
        Prices=""3131.39"";
        Volumes=""-37.84 (-1.19%)"";
};

CAC 40={
        Stocks=""CAC 40 "";
        Prices=""4265.04"";
        Volumes=""-47.26 (-1.10%)"";
};

S&P TSX={
        Stocks=""S&P TSX "";
        Prices=""15476.77"";
        Volumes=""-48.05 (-0.31%)"";
};

NUM_RECORDS=3;
```

- In case the DataField is of type **DataFieldArray**, then the output looks like:

```
Indexes[0-8]=""EURO STOXX 50 "",""3131.39"",""-37.84 (-1.19%)"",""CAC 40
",""4265.04"",""-47.26 (-1.10%)"",""S&P TSX "",""15476.77"",""-48.05
(-0.31%)"";
NUM_RECORDS=9;
```

- If the user selects to output the data to **Shared Memory** then the following value is dumped into the selected variable:

```
EURO STOXX 50|3131.39|-37.84 (-1.19%)|CAC 40|4265.04|-47.26 (-1.10%)|S&P TSX|
15476.77|-48.05 (-0.31%)
```

The output is a string result of the concatenation of all the values. The separator is by default a pipe character (|), but it can be changed.

Input Type – SQL

In this mode, the plugin reads a sheet from a database that supports SQL. The parameters in this mode are:

- **Connection String:** Connects to the database using the connection string. Each type of database has a different connection string. A good reference on connection strings for several different databases can be found in http://www.connectionstrings.com/.
- **SQL Query:** Queries the database using SQL command.
- **Key:** Determines the name of the column in the read query or table from which the keys to the DataTable should be taken.
- **From row:** Determines the row number in the query/table from where to start reading the records.
- **Number of rows:** Determines the maximum number of records to read, starting from the row stated in *From row*.
- **Fields to Read:** Specifies by name the columns to read.
- **DP Field Name:** Specifies the field name in DataPool where to write the output.
- **Field is a:** Specifies the type of the target.
- **Post Update Action:** Specifies commands (either DataPool or Viz commands) to be executed right after the data is read and sent to the target.
- **Show Data:** Dumps the data the plugin reads to a DataPool variable called READER_CONSOLE when set to On. This is extremely useful for development and debugging purposes. This dump includes error messages too.
- **Return Variables Prefix:** Specifies a prefix to add to the name of the DataPool variables that the plugin uses. For example, if the prefix is STV_, then the plugin dumps the debug information to STV_READER_CONSOLE.
- **Load Automatically:** Reads the information automatically every certain amount of time. This amount of time is defined in *Automatic Load Period (in seconds)*.
- **Automatic Load Period (in seconds):** Specifies the period of time between consecutive automatic reads.
- **Load:** Invokes the action of reading the data. The different types of outputs are exactly as in the case of Excel.

Input Type – XML

This mode allows for reading XML feeds. These can be read from a remote server or from a local file.

These can be read from a remote server or from a local file.

The parameters in this mode are:

- **File Name:** Shows the full name of a file or a URI to a remote file to be read.
- **Use Authentication:** Allows entering a user name and password In the case the *File Name* is a URI and the remote server requires Basic Authentication.
- **Avoid Cache:** Avoids using cache mechanisms that avoid the refresh of data changing in the server side in the case of a remote file. This is useful when reading information that changes. The caching mechanism avoids the plugin to get the updates. When "Avoid Cache" is on the plugin gets the freshest information.
- **XPath:** States what records to search for. For an explanation of the XPath syntax please refer to http://www.w3.org/TR/xpath/. For examples of how to use XPath please refer to https://msdn.microsoft.com/en-us/library/ms256086.
- **Namespaces:** Specifies namespaces for use in XPath expressions when it is necessary to define new namespaces externally. Namespaces are defined in the XML style, as a space-separated list of namespace declaration attributes. You can use this property to set the default namespace as well. An example of the definition of namespaces could be: `xmlns:na='http://myserver.com' xmlns:nb='http://yourserver.com'`
- **Key:** Determines the name of the field in the read records from which the keys to the DataTable should be taken.
- **From row:** Determines he row number in the read record from where to start reading the records.
- **Number of rows:** Determines the maximum number of records to read, starting from the row stated in *From row*.
- **Fields to Read:** Specifies by name the fields to read from the records.
- **DP Field Name:** Specifies the field name in DataPool where to write the output.
- **Field is a:** Specifies the type of the target.
- **Post Update Action:** Specifies commands (either DataPool or Viz commands) to be executed right after the data is read and sent to the target.
- **Show Data:** Dumps the data the plugin reads to a DataPool variable called READER_CONSOLE when set to `ON`. This is extremely useful for development and debugging purposes. This includes error messages too.
- **Return Variables Prefix:** Specifies a prefix to add to the name of the DataPool variables that the plugin uses. For example, if the prefix is `STV_`, then the plugin dumps the debug information to `STV_READER_CONSOLE`.
- **Load Automatically:** Reads the information automatically every certain amount of time. This amount of time is defined in *Automatic Load Period (in seconds)*.
- **Automatic Load Period (in seconds):** Specifies the period of time between consecutive automatic reads.
- **Load:** Invokes the action of reading the data.

Input Type – JSON

This mode allows for reading JSON feeds. These can be read from a remote server or from a local file.

The parameters in this mode are:

- **File Name:** Shows the full name of a file or a URI to a remote file to be read.
- **XPath:** States what records to search for. This is a subset of the standard XPath.
- **Key:** Determines the name of the field in the read records from which the keys to the DataTable should be taken.
- **From row:** Determines the row number in the read record from where to start reading the records.
- **Number of rows:** Determines the maximum number of records to read, starting from the row stated in *From row*.
- **Fields to Read:** Specifies by name the fields to read from the records.
- **DP Field Name:** Specifies the field name in DataPool where to write the output.
- **Field is a:** Specifies the type of the target.
- **Post Update Action:** Specifies commands (either DataPool or Viz commands) to be executed right after the data is read and sent to the target.
- **Show Data:** Dumps the data the plugin reads to a DataPool variable called `READER_CONSOLE` when set to `On`. This is extremely useful for development and debugging purposes. This dump includes error messages too.
- **Return Variables Prefix:** Specifies a prefix to add to the name of the DataPool variables that the plugin uses. For example, if the prefix is `STV_`, then the plugin dumps the debug information to `STV_READER_CONSOLE`.
- **Load Automatically:** Reads the information automatically every certain amount of time. This amount of time is defined in *Automatic Load Period (in seconds)*.
- **Automatic Load Period (in seconds):** Specifies the period of time between consecutive automatic reads.
- **Load:** Invokes the action of reading the data.

6.2.48   DataRotation

DataRotation rotates the container according to the received data.
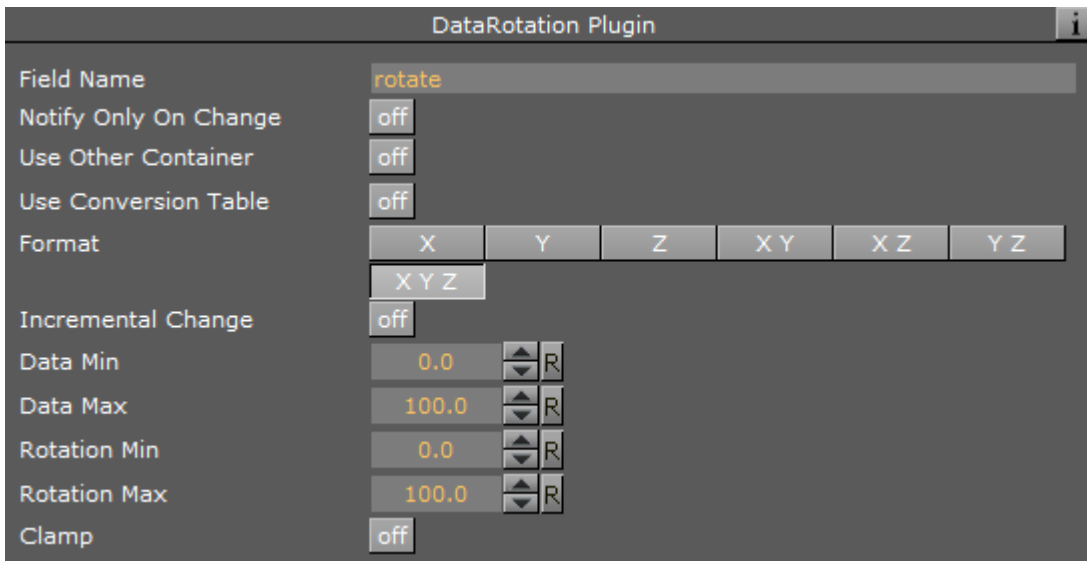
Unique Parameters

- · **Format:** Specifies the format of the incoming data. Format options specify the axis which the data relates to.
- · **Incremental change:** Defines if the received data is added to the current value of the data field. When set to `On` the received data is added to the data field. When set to `Off`, the received data replaces the value of the data field.
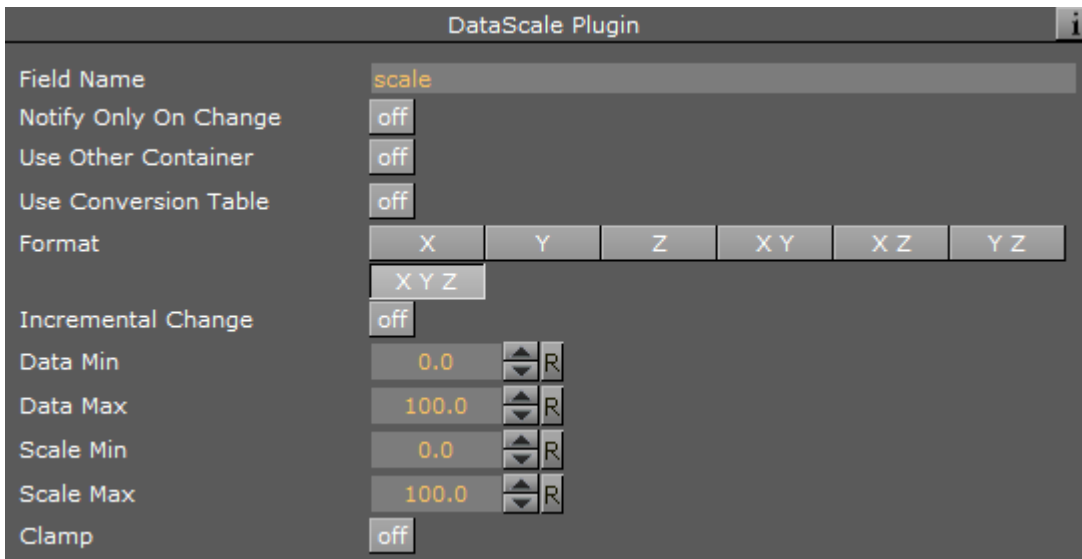
Example

If Field Name is `VALUE` and the format is XY, the data sent is `VALUE=10.5 5.0;`. The container rotates around the X axis to an angle of `10.5` and around the Y axis to an angle of `5.0`.

6.2.49   DataScale

DataScale scales the container according to the received data.

Unique Parameters

- · **Format:** Specifies the format of the incoming data. Format options specify the axis which the data relates to.
- · **Incremental change:** Defines if the received data is added to the current value of the data field. When set to `On` the received data is added to the data field. When set to `Off` the received data replaces the value of the data field.
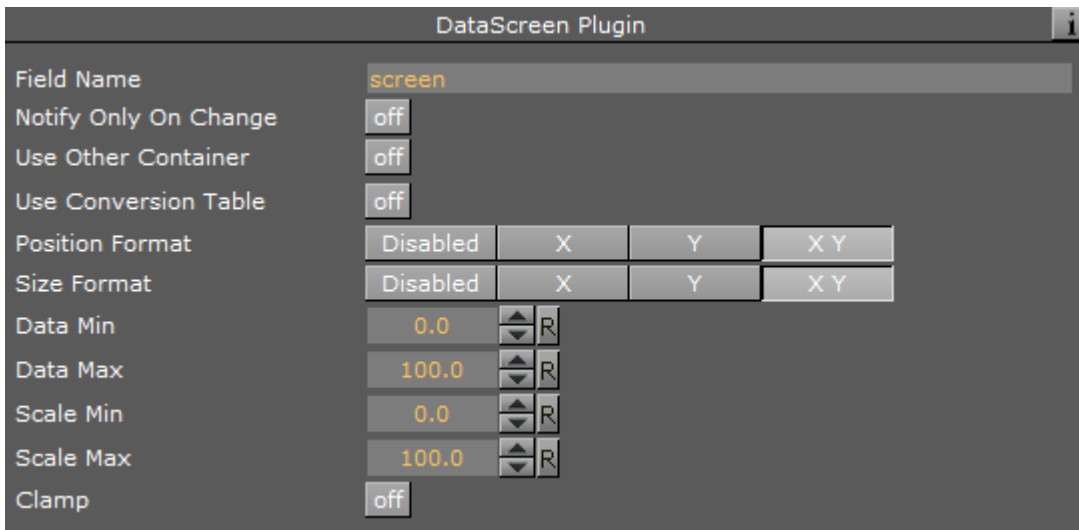
Example

The current scaling values of the container are `2.0 3.0 4.0`. If the parameters are: `Field Name: VALUE`, `Format: Y`, `Data Min: -100`, `Data Max: 100`, `Scale Min: 0` and `Scale Max: 1` and the data entered is `VALUE=0;`. The resulting scaling values of the container are `2.0 0.5 4.0`

6.2.50 DataScreen

☐
DataScreen is used to control screen coordinates transformations of objects.

Unique Parameters

- **Position Format:** Defines if the object's position is affected by incoming data and the format the data. Data is used as screen coordinate values, where the top left corner of the render window is the 0 0 position and the bottom right corner of the render window is 720 576 position.
- **Size Format:** Defines if the object's scaling is affected by incoming data and the format of the data.

6.2.51   DataScript

DataScript plugin extends DataPool capabilities by enabling the use of a scripting tool. DataScript uses JavaScript format with special API to DataPool variables.

> **IMPORTANT!** The DataScript plugin is NOT SUPPORTED for real-time broadcast graphics and it should not be used in any scene that is used to go On Air.

For detailed information about the DataScript plugin refer to the DataScript user manual.
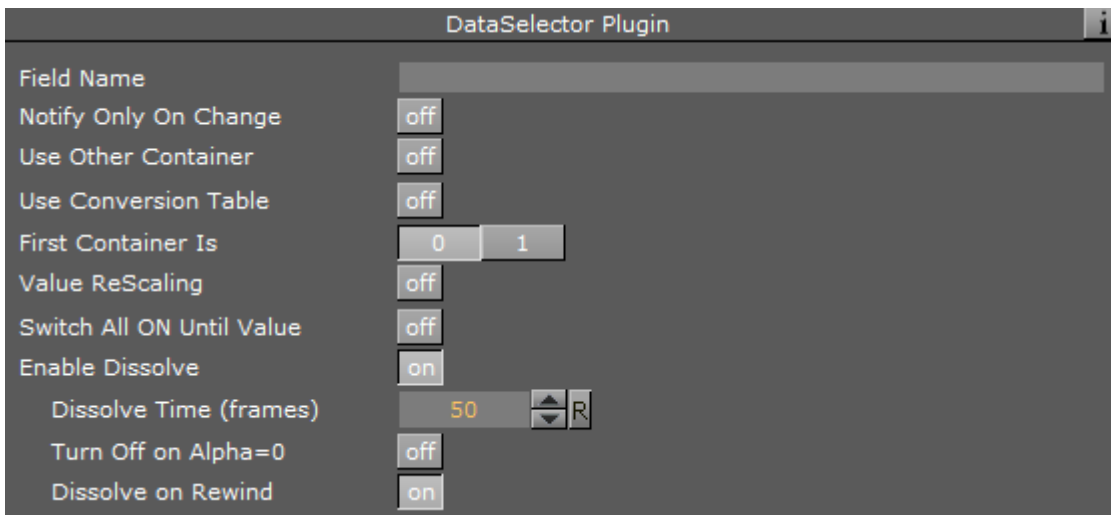
Unique Parameters

- **Data Contains Script:** Defines whether incoming data is interpreted as a script or as regular data.
- **Execute On Load:** Defines if the script runs when the scene is loaded to Viz or waits until script execution is triggered.
- **Execute:** Executes the script when pressed.

6.2.52 DataSelector

☐

DataSelector receives a number of a child container as data. The plugin toggles visibility for all the child containers of the controlled container and activates only the *n*-th child container, as received in the data.

Unique Parameters

- **First Container Is:** Select 0 or 1 as the first container index. Incoming data is applied accordingly.
- **Value ReScaling:** Disables "First Container is" parameter and enables additional parameters when set to On, allowing rescaling of incoming data to the defined values. All incoming data exceeding the Data Min or Data Max is cropped.
- **Switch all on until value:** Switches on visibility for all child containers until the first value is accepted in the DataField when set to On. After the first value is set, the DataSelector only switches one child container at a time. When set to Off, child containers keep their current status until the first value is accepted in the DataField.
- **Enable Dissolve:** Uses a fade transition between the current container and the next container.

Example

Field Name=VALUE. If the data entered is VALUE=3 then all the children containers are invisible out of the fourth (0-3) child.

6.2.53 DataSHM

☐

The DataSHM (Shared Memory) plugin uses a defined data field or a DataPool expression to update a string type shared memory entry.

> ⚠ **Note:** This plugin exists for Viz 3 only.

Unique Parameters

- **Shares Memory Type:** Selects the shared memory in Viz 3 that the memory key is defined in. For additional information about Viz 3 shared memory, refer to Viz 3 user guide.
- **Shares Memory Key Name:** Defines the shared memory key name that receives the data field value or DataPool expression.
- **Transferred Argument:** Selects the argument type that is sent to the shared memory key. Select **Variable Value** to set the data field value to the memory key. Select **DP Expression** to send a DataPool expression to the shared memory key. When DP Expression is selected, an additional parameter is enabled: Expression String. Define the DataPool expression that results in a string. The resulting string is sent to the shared memory key.

6.2.54   DataSHMTracker

The DataSHMTracker plugin monitors a string type shared memory entry. When the shared memory entry is changed, the plugin uses the updated value to update the related DataPool variable.

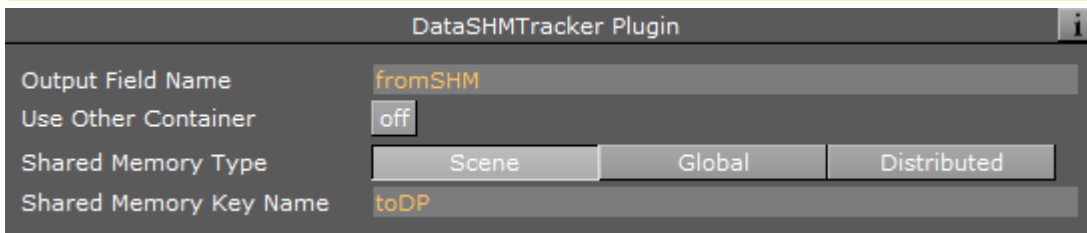> ⚠ **Note:** This plugin exists for Viz 3 only.

Unique Parameters

- · **Shares Memory Type:** Selects the shared memory in Viz 3 that the memory key is defined in. For additional information about Viz 3 shared memory, refer to Viz 3 user guide.
- · **Shares Memory Key Name:** Defines the tracked shared memory key name. Whenever the shared memory key changes, its value is copied to the defined data field.

6.2.55 DataStructure

☐

DataStructure plugin uses structures that were defined in the DataPool scene plugin. The structure format is the same as in the *config.dp* file but the structure is defined in the scene making it available only to variables in the scene (Unlike structures defined in the *config.dp* file which are available for all scenes to use). The Field name relates to a variable defined as the type of a structure.
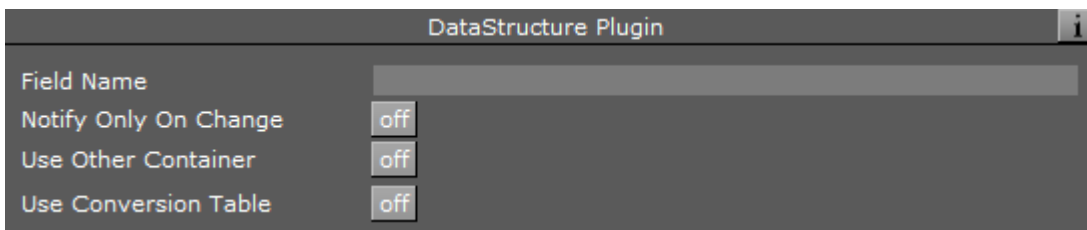
6.2.56 DataSwitch

☐

DataSwitch toggles the visibility of the container according to the incoming data. Any value other than zero sets the container's visibility to `On`, zero sets the visibility to `Off`.
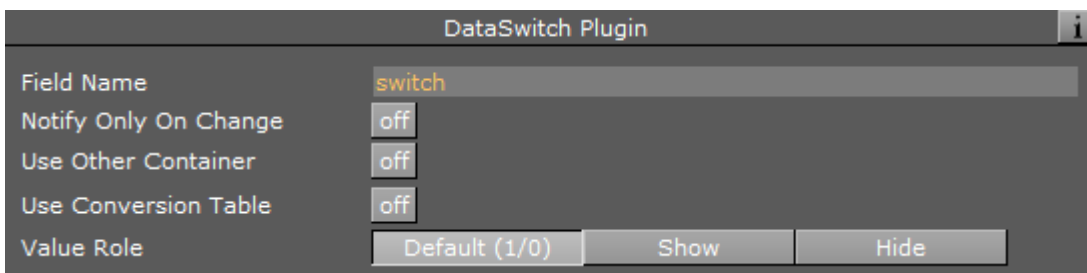
Example

`Field Name=VALUES[3]`. If the data entered is `VALUES[0-2]=0, 1, 0;`, then the first and third children of the container the plugin is attached to is deactivated and the second child is activated. If the data entered is `VALUES=0`, the container itself is deactivated. If the data is `VALUES=1`, the container itself is activated.

6.2.57 DataSystem

DataSystem plugin enables running external applications/commands. Incoming data contains the command to run.
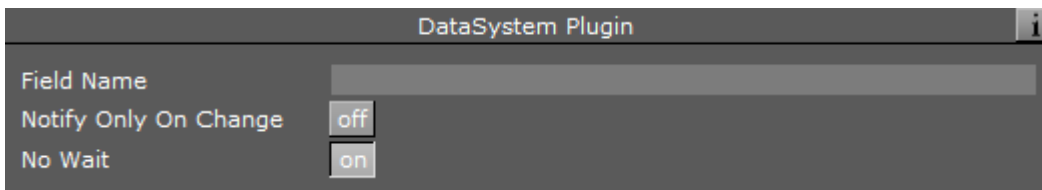
Unique Parameters

· **No Wait:** Defines if Viz waits for the launched application to finish its run or not.

Example

Field Name is CMD. Incoming data is CMD=`calc;`. The result is that the windows calculator opens.

6.2.58 DataTable

DataTable defines a table of typed objects.

Unique Parameters

- **Table Name:** Defines the name of the table.
- **Type:** Defines the type of the objects in the table. The list of the types is taken from the configuration file.

6.2.59 DataText

☐

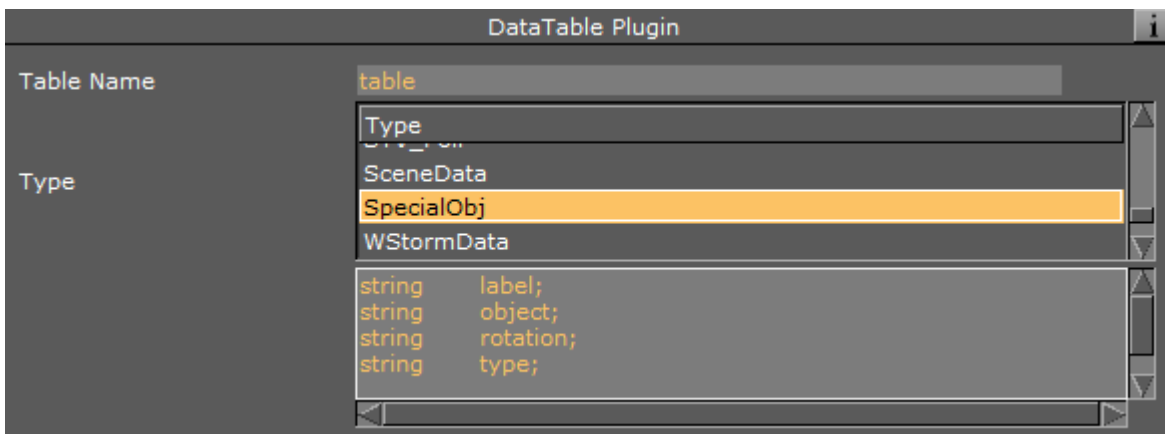DataText controls text objects. The plugin receives incoming data, adds a defined prefix and a defined suffix to the data, and changes the text value (string) to the result.

DataText Plugin					i
Field Name	text				
Notify Only On Change	off				
Use Other Container	off				
Use Conversion Table	off				
Prefix					
Suffix					
Rebuild Now	0	R			
Trim Options	on				
No Delimiter -> Empty String	off				
Remove Prefix Until (including)	on				
Sub String	[				
Remove Suffix From (including)	on				
Sub String	]				
Choose an Entry	on				
Delimiter	::				
Number	1	R			
Choose Characters	on				
Range	1-5				
Choose Lines	on				
Range	2-3				
Replace back slash n by EOL	off				
Replace back slash Quotation by Quotation	off				
Convert Case	None	Lower	Upper	Word	Sentence

Unique Parameters

- **Trim Option:** Enables/disables trimming. When set to `Off`, incoming data is not trimmed. When set to `On`, additional trim options for incoming data are added:
  - **Remove Prefix Until:** Removes all characters from the beginning of the string to the defined delimiting substring (including) when set to `On`. An additional parameter is enabled: Sub String, to define a limiting substring.

- **Remove Sufix From:** Removes all characters from the defined substring (inclusive) to the end of the string when set to `On`. An additional parameter is enabled: Sub String, to define the substring.
- **Choose an entry:** Enables additional parameters when set to `On`.
- **Delimiter:** Defines a delimiting character.
- **Number:** Defines the entry number of the beginning with the delimiter character. The incoming string is split into substrings, using delimiter Y as the split point (end of sub string), and substring X is used as the data (delimiting characters are not included in the substrings).
- **Choose Characters:** Enables the Range parameter when set to `On`. The data used by the plugin is a simple range of bytes X-Y defined in Range (zero is not a valid value as a character number).

> ⚠ **Note:** When using more than one trim option the AND operator is used, i.e., if all options are used the following result is used: Remove prefix AND remove suffix AND split data AND select substring number X AND select bytes number X-Y.

- **Replace backslash n by EOL:** Enables/disables replacement of control characters. When set to `Off`, incoming data is not changed. when set to `On`, all `\n` (backslash n) control characters are replaced with the End Of Line control character.
- **Convert Case:** Selects the required option for data case conversion:
    - **None:** Does not change any data strings.
    - **Lower:** Converts all data strings to lower case only.
    - **Upper:** Converts all data strings to upper case only.
    - **Word:** Converts the first letter of every word to upper case.
    - **Sentence:** Converts the first letter of every sentence to upper case.

Example

Field Name=`PRICE`
Prefix= `$`
Suffix= `is the price.`

If the data is PRICE=123 then the result is: `$123 is the price`.

> ⚠ **Note:** When sending special characters, or characters used as DataPool separator characters, to *DataText* plugin, use double-double quotes at the beginning and ending of the string: ""*What's the frequency, Kenneth?*"" If the quotes are omitted, the string is not displayed correctly.

6.2.60 DataTextKerning
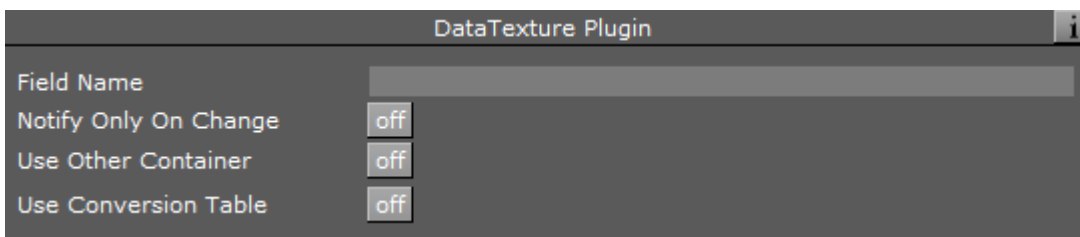
DataTextKerning sets the kerning of a text object.
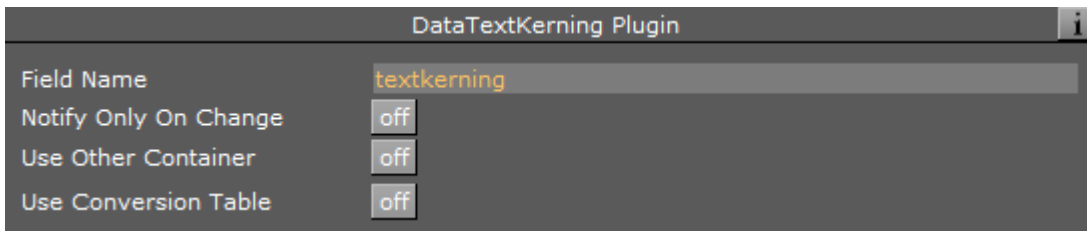
Example

Field Name=VALUE. If the entered data is VALUE=`12`; then the kerning of the text object is assigned the value `12`.

6.2.61   DataTexture

☐

DataTexture controls various texture attributes such as mapping, scaling, etc.

Incoming data contains the parameter name of the texture and the value.

Supported Parameters

- **TEXT:** Creates a texture on the controlled container. Value is `IMAGE*<image name>` where image name is a full path to an image in Viz images library.
- **MAPT:** Defines the mapping type of the texture. Values are `1` (LINEAR), `2` (VERTEX), `3` (REFLECT).
- **ENVT:** Defines the environment type of the texture. Values are `3042` (blend), `8449` (decal) or `8448` (modulate).
- **QUAL:** Defines the texture quality type. Values are `1` (PIXEL), `2` (LINEAR), `3` (MIPMAP), `4` (SHARPEN).
- **COLO:** Defines the color quality of the texture. Values are `1` (32 bits), `2` (16 bits).
- **EFFT:** Defines an effect type used on the image. Values are `0` (Smooth), `1` (Mosaic).
- **WRAP:** Defines the wrapping type of the texture. Values are `10497` (REPEAT), `10496` (CLAMP).
- **POSX:** Position of the texture on the X axis of the container. Value should be a float number.
- **POSY:** Position of the texture on the Y axis of the container. Value should be a float number.
- **ROTZ:** Rotation of the texture around the Z axis. Value should be float number.
- **SCAX:** Scales texture on the X axis. Value should be float number.
- **SCAY:** Scales texture on the Y axis. Value should be float number.
- **BLEN:** Provides blend value of the texture. Value should be a float number.

- **EFFV:** Defines the amount of the effect defined in the `EFFT` (effect type) that is applied to the texture. Value should be a float number.
- **IMGS:** Changes the image of a given container. Value is *<Xvalue> <Yvalue>*.
- **SETI:** Replaces the image on the container. Data format is `IMAGE*<image Name>` where image name is the full path to an image in the Viz images library.
- **DRAW:** Sets the scale and bias values of the texture. Values are *<texcoord scale> <float bias s> <float bias t>*. all values are float numbers.

6.2.62   DataTime

☐

DataTime plugin translates the value of the incoming data into date/time elements and stores these values to a set of predefined DataPool variables.

Predefined Variables

The predefined variables are:

- **_DAY:** Day of the month (`1-31`).
- **_DAYNAME:** Name of the day in the week. Names are defined in the Day Names parameter.
- **_FHOUR:** Hours/min/sec described in floating point number (decimal) i.e. `6:30` is converted to `6.5`.
- **_HOUR:** Hour of the day (`0-23`).
- **_HOURNAME:** Set of name description for the hours of the day. Names are defined in the Hour Names parameter.
- **_MIN:** Minute of the hour.
- **_MONTH:** Month number in the year (`1-12`).
- **_MONTHNAME:** Name of the month. Names are defined in the Month Names parameter.
- **_MSEC:** One thousandth of a second in the second (`0-999`). This variable is used only when the Resolution parameter is set to milliseconds.
- **_SEC:** Second number in a minute (`0-59`).
- **_WEEKDAY:** Number of the day in the week (`1-7`).
- **_YEAR:** Number of the year using four digits.
- **_RELDAY:** Relative day number, where `0` (zero) is today. This variable returns an integer value only (i.e. the difference between the received data and the current system time).
- **_RELDAYNAME:** Relative day name, where yesterday, today and tomorrow are used for `-1`, `0`, `1` values received in the `RELDAY` variable. All other days are displayed by name (Sunday, etc.).
- **_HOURS_SHORT:** Show 12 hour clock time.
- **_AMPM:** Display AM or PM (used with `HOURS_SHORT`).
  ☐

Unique Parameters

- **DP variables Prefix:** Defines a prefix that is added to the DataTime variables. The prefix is used to distinguish one set of time parameters from the other, enabling the usage of multiple time values. Default is `T1`. If the prefix is set to `T1`, the corresponding variables are:
    - T1_DAY
    - T1_DAYNAME
    - T1_FHOUR
    - T1_HOUR
    - T1_HOURNAME
    - T1_MIN
    - T1_MONTH
    - T1_MONTHNAME
    - T1_MSEC
    - T1_SEC
    - T1_WEEKDAY
    - T1_YEAR
- **DP variables Scope:** Uses data variables as part of the structure the variables were defined in when set to `Local`. When set to `Global`, the time data variables are used as global DataPool variables.
- **Input is Machine Time:** Defines if the time is read from the machine's clock or from the data field. If set to `On`, the Field Name parameter is disabled and additional parameters are enabled:
- **Input Field Format:** Selects the input field units.
- **Output Field Time Zone:**
    - **As Is:** Leaves timezone data as set.
    - **UTC -> Local:** Converts UTC timezone to local timezone.
    - **Local -> UTC:** Converts local timezone to UTC.
- **Offset Field:** Defines the offset units.
- **Offset:** Defines the number of offset units from the current machine time.
- **Leading Zero:** Defines if the zero character is added to the single digit values (`0-9`). This option is useful when formatting the time display.
- **Day Names:** Defines a set of names for the days of the week, to be used by the `_DAYNAME` variable. The parameter should contain seven names, comma separated. If the parameter is left blank, *DataTime* uses the default of Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday.
- **Month Names:** Defines a set of names for the months of the year, to be used by the `_MONTHNAME` variable. The parameter should contain twelve names, comma separated. If the parameter is left blank, *DataTime* uses the default of January, February, etc.
- **Hour Names:** Defines a set of names for the hours of the day, to be used by the `_HOURNAME` variable. The parameter should contain twenty four names, comma separated. If the parameter is left blank, *DataTime* uses the default of Midnight, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, Noon, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23.

- **Replace Yesterday By:** Allows the replacement of the term *yesterday* for another one. This feature is useful when working on a language other than English.
- **Replace Today By:** Allows the replacement of the term *today* for another one. This feature is useful when working on a language other than English.
- **Replace Tomorrow By:** Allows the replacement of the term *tomorrow* for another one. This feature is useful when working on a language other than English.
- **Initialize:** Calculates variable values now.
- **Reset:** Recalculates variable values now.
- **Week Director**

6.2.63   DataTimer

DataTimer plugin runs an action (Viz action or DataPool action) at a fixed time interval.
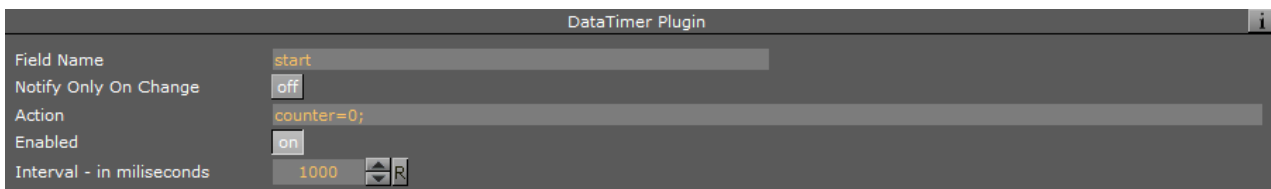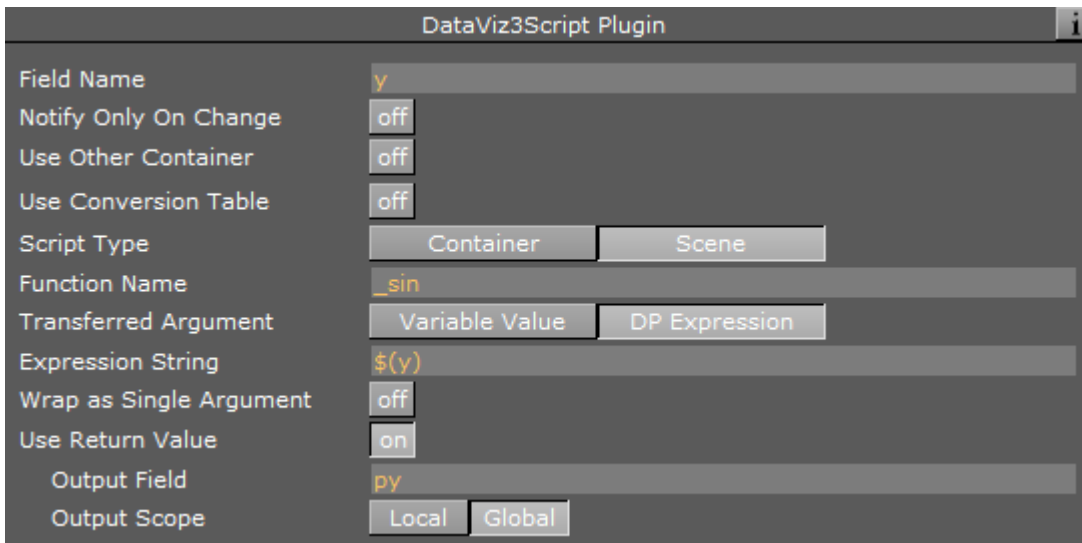

Unique Parameters

- **Action:** Defines the action that is executed at the interval time.
- **Enabled:** Executes the action at the defined time intervals when set to `On`. When set to `Off` the plugin is disabled and the action is not executed.
- **Interval:** Defines the time length, in milliseconds, between executions of the action.

6.2.64   DataViz3Script

DataViz3Script plugin invokes a predefined Viz 3 scripting subroutine or function, with the updated value of its defined data field (DataPool variable) or with a DataPool expression as an argument. In the case of a function, the returned value of the function can be assigned to a DataPool variable.

> ⚠ **Note:** This plugin exists for Viz 3 only.

Unique Parameters

- **Script Type:** Defines the location of the script, containing the requested function/procedure. Select **Container** to use a function in a container script or **Scene** to use a function in the scene setting script module.
- **Function Name:** Sets the name of the function/procedure to be called when the data field changes.
- **Transferred Argument:** Selects the argument that is sent to the function/procedure. Select **Variable Value** to set the data field value to the memory key and to use the data field value as the function argument. Select **DP Expression** to send a DataPool expression to the function/procedure. When DP Expression is selected, an additional parameter is enabled: Expression String. Define the DataPool expression that results in a string. The resulting string is sent as an argument (or arguments) to the function/procedure.
- **Use Return Value:** Defines if the returned value from the defined function is re-used by the plugin. When set to `Off`, no return value is used. When set to `On`, additional parameters are enabled:
    - **Output Field:** Sets the data field name that the returned value is assigned to.
    - **Output Scope:** Defines the output field scope: Local or global. The value returned from the function is assigned to the defined output field.

6.2.65   DataWPosition

☐

DataWPosition plugin affects the WPosition plugin (a Viz Weather plugin) attached to the controlled plugin. The DataPool variable defined in the Field Name should contain the values of the Longitude, Latitude and Altitude to be sent to the WPosition plugin.