Data Object

PRIME V.4.9

Contents

Introduction 3
Basics 5
Supported Data Sources 6
- Text 7
- XML 8
- RSS 9
- Web 10
- JSON 12
- Excel 13
- Access 14
- DSN 15
- AP Web Feed 16
- Images 16
- Movies 17
- Google 18
- ChyronHego Data Engine 19
- Parameters 19

Filtering 19
- All Rows 20
- All Rows Where Column 20
- Now Next Later 21
- Custom Query 21
- Parameters in Query Strings 23

Binding Types 23
- Tree Binding 24
- Cell Binding 25
- Column Binding 25
- Table Binding 27
- How to Bind Targets 29
  - Drag and Drop 29
- Formatting Data 35
  - Text 35
  - Number 36

Date 38
Time 38
Commands 41
Timeline 47
Timer 47
Events 49

Introduction

The Data Object is used in order to connect a data source and binding portions of a data set to object properties within a scene. Prior to its application, the data may be filtered/formatted using easily configurable options within the Data Object editor.

Although the Data Object existed in previous versions of Channel Box, the implementation in PRIME differs greatly. The configuration editor is presented as an independent panel that may be docked or repositioned anywhere in the editor. Furthermore, the panel now presents a current view of the last selected Data Object and any changes are committed to that Data Object immediately.

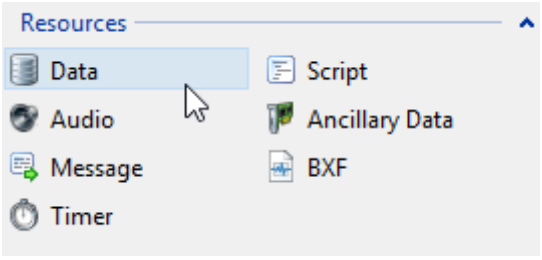
Most importantly, instead of binding from a preset list of objects, the user may now drag-and-drop objects and properties directly onto the Data Object binding section, and bindings support the same expression functionality available elsewhere in PRIME.

One crucial difference, between binding using the Data Object and binding in other aspects of PRIME is that binding with the Data Object is centered across the execution of commands. With other bindings, as a value changes, the new value is immediately pushed out to any of the bound targets. To the contrary, the Data Object only pushes values to bound targets when an Update command instructs it to do so.

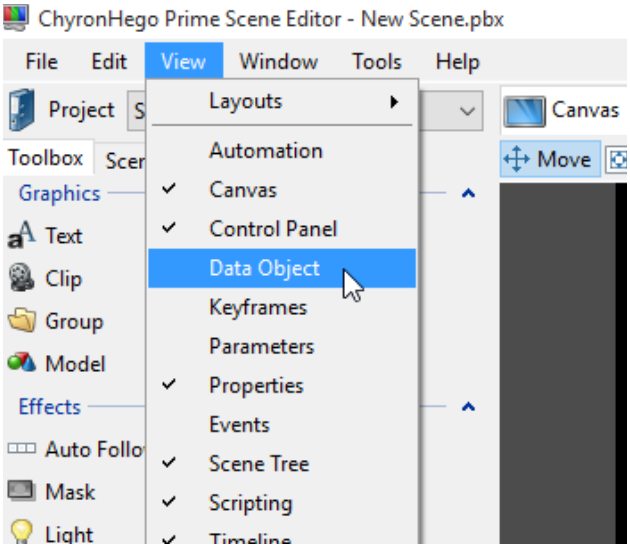
Basics

To add a new Data Object to a scene:

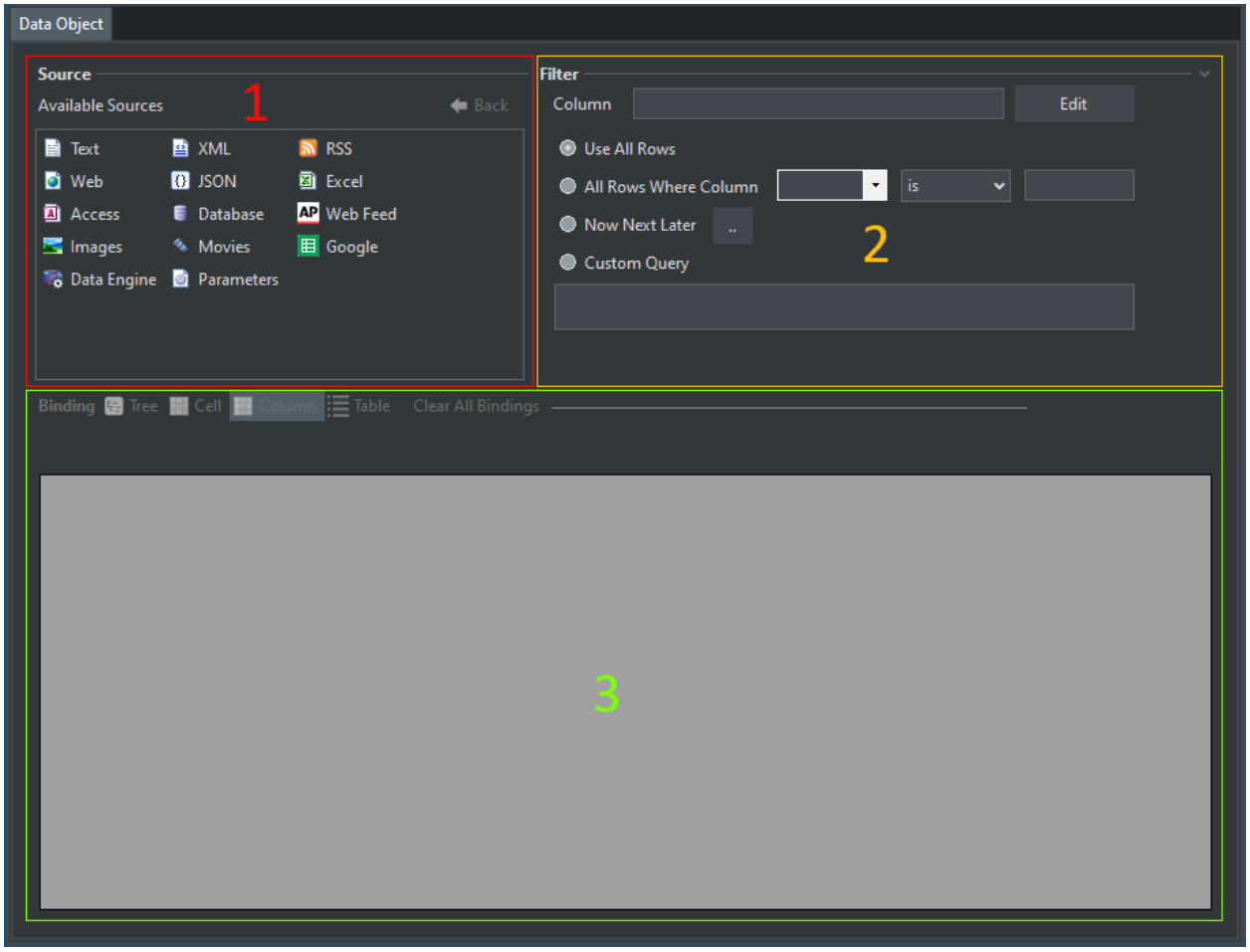
- Click **Data** under the Resources section of the scene toolbox. Doing so will insert and select the new object, which is visible in the scene tree under the Resources section as well.



Whenever a Data Object is selected, the configuration panel will automatically appear, if currently hidden. The panel may also be hidden or displayed through the Data Object option in the View menu.



By default, the Data Object configuration panel will appear docked next to the Canvas, Control Panel and Scripting tabs.



As seen above, the panel is functionally broken down into three sections:

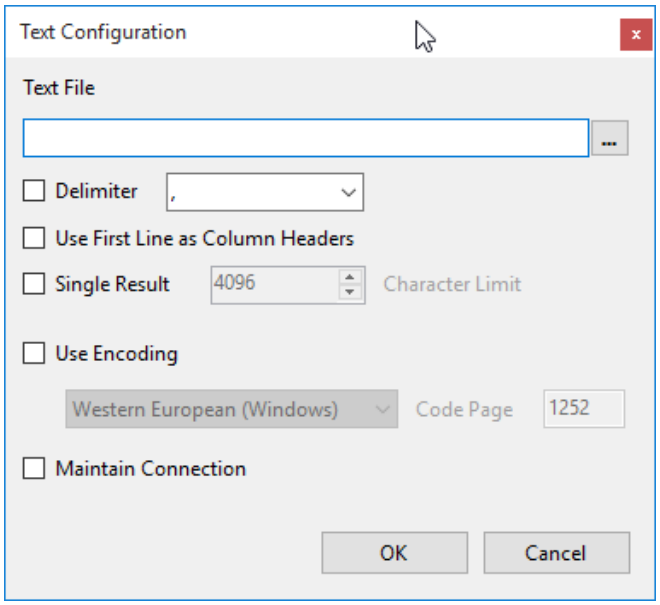
1. **Source** – First, the user chooses one of the available data sources and then connects to a specific data set.
2. **Filter** - Once connected, the user may choose to filter the data set using one of the supported filter types. Not all filters are supported by all data sources; unsupported filters are automatically disabled.
3. **Binding** - The binding section presents a preview of the selected data set. The user may then bind to individual sections of the data, depending contextually on which binding types are supported by the data source and which binding type is currently active.

Supported Data Sources

Each data source type has a unique collection of properties that must be defined in order to interact with a data set.

Text

The Text data source allows the Data Object to connect to a plain text file. Typically, the file will be on the local file system. However, a network path (UNC) is also supported. Once selected, the following dialog will appear with configuration options relevant to the Text data source.



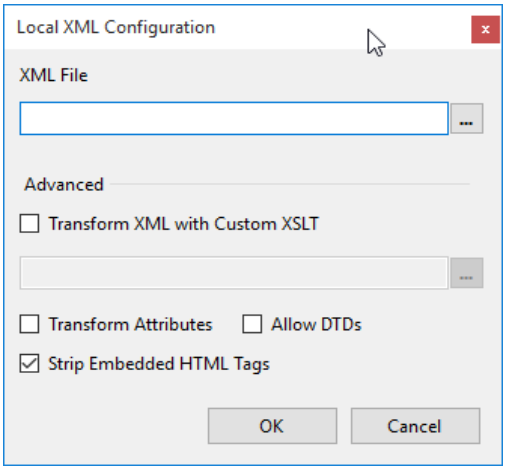
- **Text File** – Specifies the file will be used as the data source. Typically, such files have the CSV, TXT or LOG extensions. Non-standard extensions are also supported.
- **Delimiter** – Specifies whether each line of the text file should be treated as delimited data. If enabled, each line will be split into multiple data fields using the configured delimiter character.
- **User First Line as Column Headers** – Determines whether the first line of text is interpreted as part of the data set or as metadata that defines field names.
- **Single Result** – Allows the entire text file to be interpreted as a single data field, although it is limited to a maximum number of characters to avoid performance issues. The **Use Encoding** property can be used to override the default settings.
- **Use Encoding** – When a text file is selected, the Data Object will attempt to identify the text encoding used in saving the file. This is necessary to ensure proper handling of Unicode text. The **Use Encoding** property may be used to override the default setting.

- **Maintain Connection** – Preserves the connection to the text file, ensuring better performance, but preventing concurrent access to the file. In practice, this means that the file will not be editable while the Data Object is using it. By default, this property should be disabled.

Note – The Text data source requires either Microsoft Office or, the freely available, Microsoft Access Database Engine. Without either installed, this data source will be unable to read text files.

XML

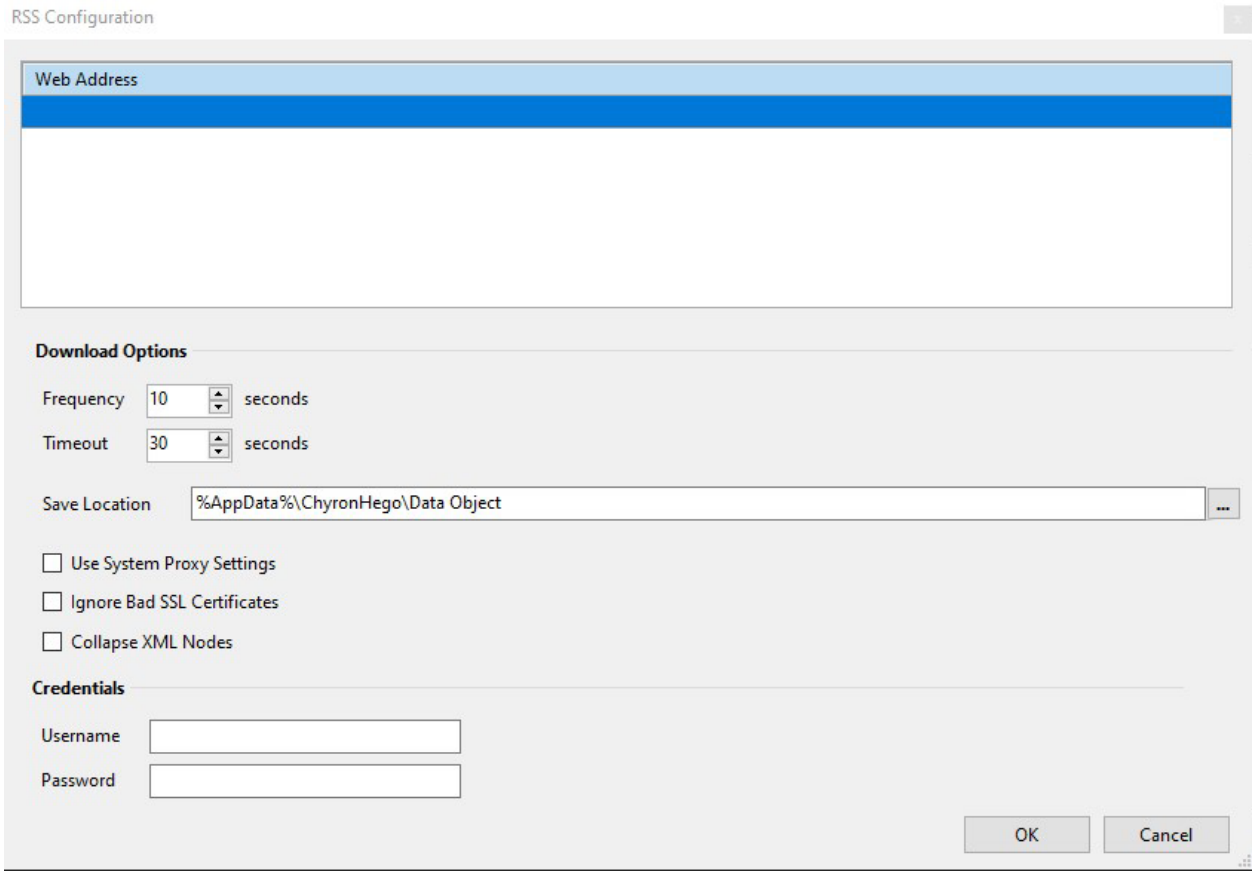
The XML data source allows the Data Object to connect to an XML file. Typically, the file will be on the local file system. However, a network path (UNC) is also supported. Once selected, the following dialog will appear with configuration options relevant to the XML data source.



- **XML File** – Defines the file that will be used as the data source. Typically, such files use the XML or RSS extension. However, non-standard extensions are also supported. XML files locating using Web addresses should use the Web or RSS data source type instead.
- **Transform XML with Custom XSLT** – Allows the user to specify a separate XSLT file, which transforms the provided XML into a different and more useful form prior to processing by the Data Object.
- **Transform Attributes** – If checked, this option will convert XML attributes into XML nodes. This may offer better results for XML files with data that is largely contained in attributes. XPath queries will need to be modified accordingly. This option is typically only needed when binding in either Cell or Column mode.
- **Allow DTDs** – Determines whether document type definition (DTD) validation rules should apply. In most cases, this option should be disabled.
- **Strip Embedded HTML Tabs** – If enabled, this option will remove HTML formatting tags from each data field. This is particularly useful when working with RSS data.

RSS

The RSS data source allows the Data Object to connect to one or more remote RSS feeds. Once selected, the following dialog will appear with configuration options relevant to the RSS data source.

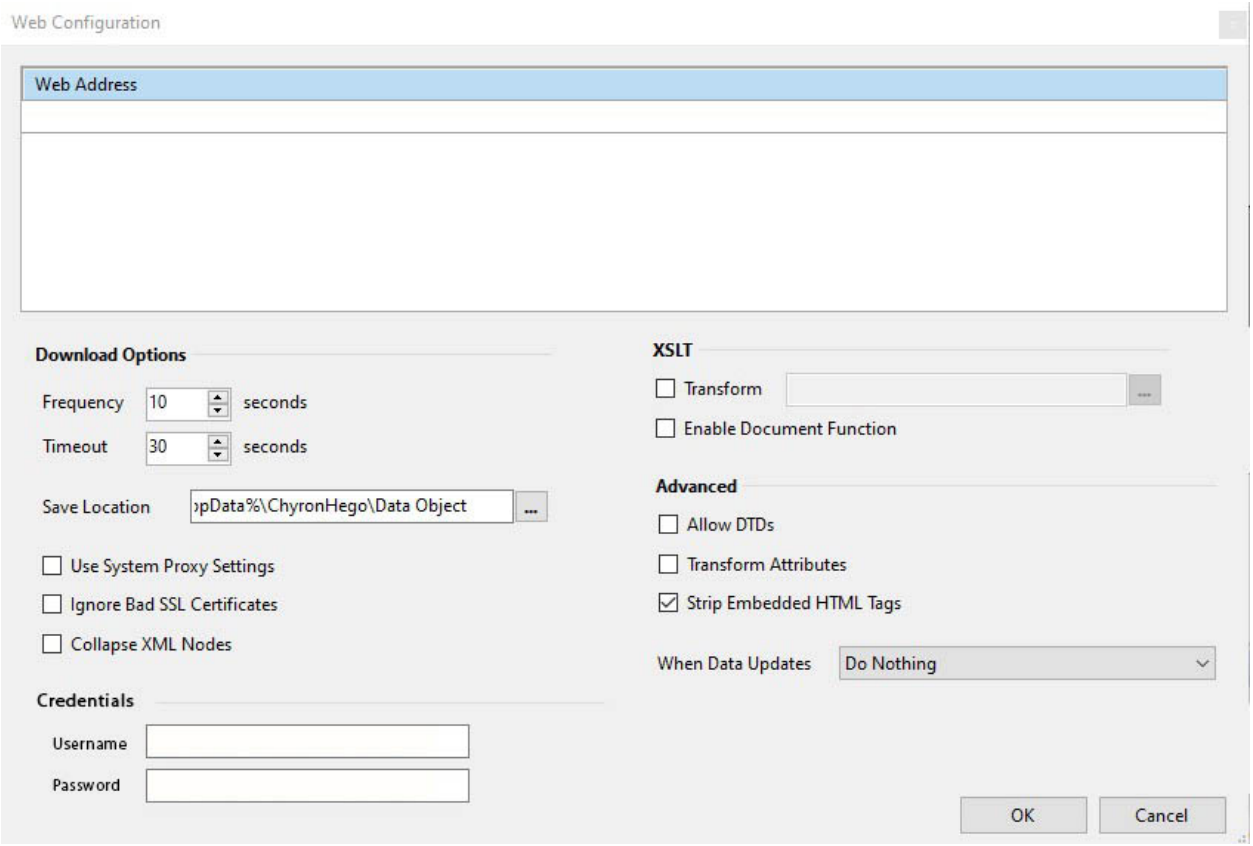


- **Web Address** – Allows the user to enter one or more URLs. Each URL should point to an RSS feed. If multiple feed URLs are specified, the resulting data set will contain data from all URLs.
- **Download Frequency** – Defines the interval in which web data is to be downloaded regularly.
- **Download Timeout** - Interval in seconds that no web data is received before download is canceled.
- **Save Location** - Web data is stored on the local file system in the *Save Location.*
- **Use System Proxy Settings** – If enabled, the proxy settings configured under the Windows Internet Options widget will be observed.

- **Ignore Bad SSL Certificates** - HTTPS web addresses with invalid SSL certificates will continue to load.
- **Collapse Node** – If checked, this option will attempt to collapse an entire XML document into a single record. This option is only appropriate when binding in either Cell or Column mode.

Web

The Web data source allows the Data Object to connect to one or more remote XML-based feeds. This data source is **NOT** used to scrape arbitrary Web pages. Once selected, the following dialog will appear with configuration options relevant to the Web data source.

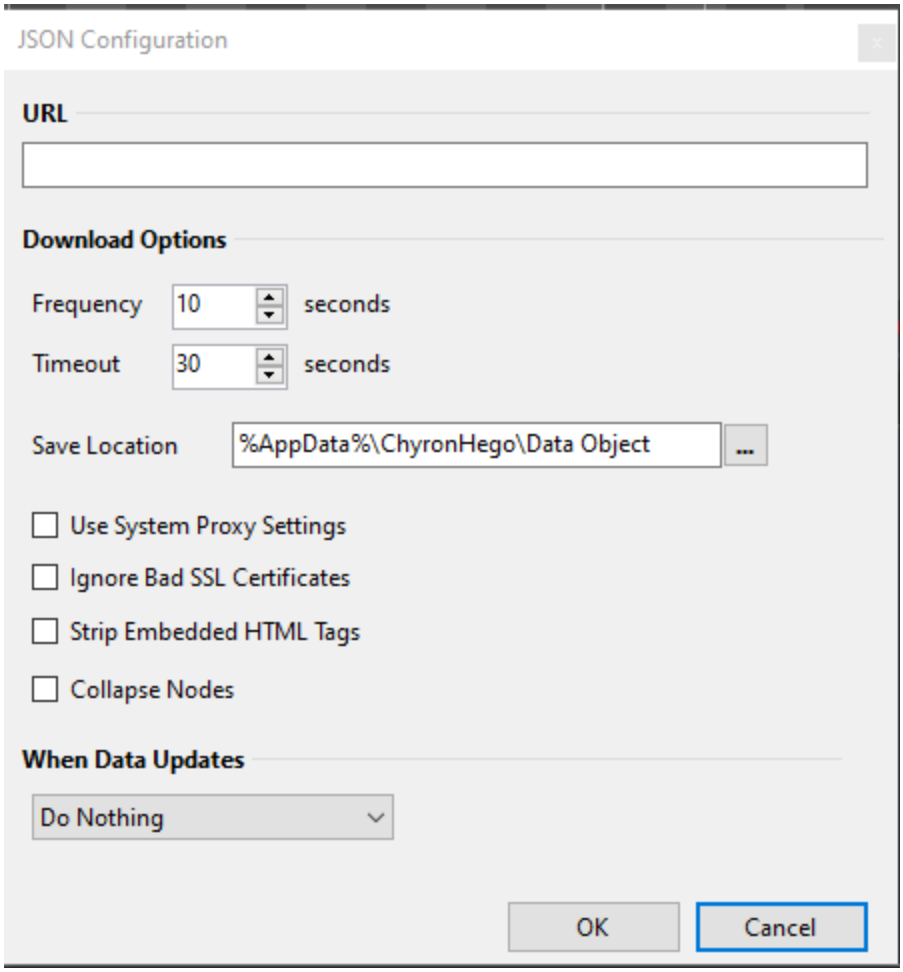


- **Web Address List** – Allows the user to enter one or more URLs. Each URL should point to an XML feed. If multiple feed URLs are specified, the resulting data set will contain data from all URLs. If the XML schema for each URL is the same, then the data will co-exist.
- **Download Frequency** – Defines the interval in which web data is to be downloaded regularly.

- **Download Timeout** - Interval in seconds that no web data is received before download is canceled.
- **Save Location** - Web data is stored on the local file system in the *Save Location.*
- **Use System Proxy Settings** - If enabled, the proxy settings configured under the Windows Internet Options widget will be observed.
- **Ignore Bad SSL Certificates** - HTTPS web addresses with invalid SSL certificates will continue to load.
- **Collapse Node** – If checked, this option will attempt to collapse an entire XML document into a single record. This option is only appropriate when binding in either Cell or Column mode.
- **Credentials** - Allows a user to input username and password credentials for a web address URL that requires authorization. These are securely stored and are used for all web address URLs when requesting xml/rss data with per data object.
- **Transform SML with Custom XSLT** – Allows the user to specify a separate XSLT file. This transforms the provided XML into a different and more useful form prior to processing by the Data Object.
- **Allow DTDs** – Determines whether document type definition (DTD) validation rules should apply. In most cases, this option should be disabled
- **Transform Attributes** – If checked, this option will convert XML attributes into XML nodes. This may offer better results for XML files with data that is largely contained in attributes. XPath queries will need to be modified accordingly. This option is typically only needed when binding in either Cell or Column mode.
- **Strip Embedded HTML Tags** – If enabled, the proxy settings configured under the Windows Internet Options widget will be observed.
- **When Data Updates** - Select the behaviour ; Do Nothing, Push Data on Change or Push Data on update.

JSON

The JSON data source allows the Data Object to connect to one or more remote JSON data sets. Once selected, the following dialog will appear with configuration options relevant to the JSON data source.

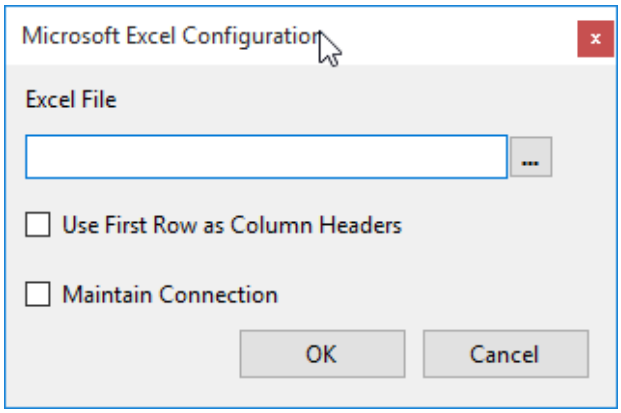


- **URL** - Defines the remote location of the desired JSON data set.
- **Download Interval** – Defines the interval in which web data is to be downloaded regularly. This data is stored on the local file system in the *Save Location.*
- **Use System Proxy Settings** – If enabled, the proxy settings configured under the Windows Internet Options widget will be observed.
- **Ignore Bad SSL Certificates**– Allows you to connect to "insecure" SSL connections and skip SSL certificate checks.

- **Strip Embedded HTML Tags** – If enabled, this option will remove HTML formatting tags from each data field. This is particularly useful when working with RSS title and description data.
- **Collapse Nodes**- allows you to expand or collapse root nodes of JSON data, to display merged or separate columns within the data object table. Improves the usability and ease of navigating to the JSON data you need within Prime.

Excel

The Excel data source allows the Data Object to connect to a Microsoft Excel file. Typically, the file will be on the local file system. However, a network path (UNC) is also supported. Once selected, the following dialog will appear with configuration options relevant to the Excel data source.

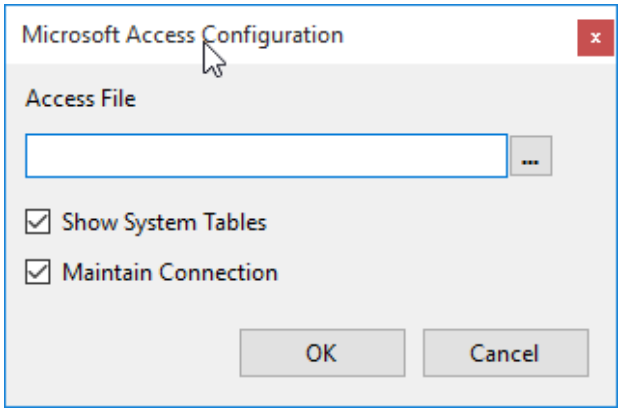


- **Excel File** - Defines which XLS or XLSX file will be used as the data source.
- **Use First Row as Column Headers** – Determines whether the first row of data is interpreted as part of the data set or as metadata that defines field names.
- **Maintain Connection** - Will preserve the connection to the Excel file, ensuring better performance but preventing concurrent access to the file. In practice, this means that the file will not be editable in Excel while the Data Object is using it. By default, this property should be disabled.

Note – The Excel data source requires either Microsoft Office or, the freely available, Microsoft Access Database Engine. Without either installed, this data source will be unable to read text files.

Access

The Access data source allows the Data Object to connect to a Microsoft Access file. Typically, the file will be on the local file system. However, a network path (UNC) is also supported. Once selected, the following dialog will appear with configuration options relevant to the Access data source.

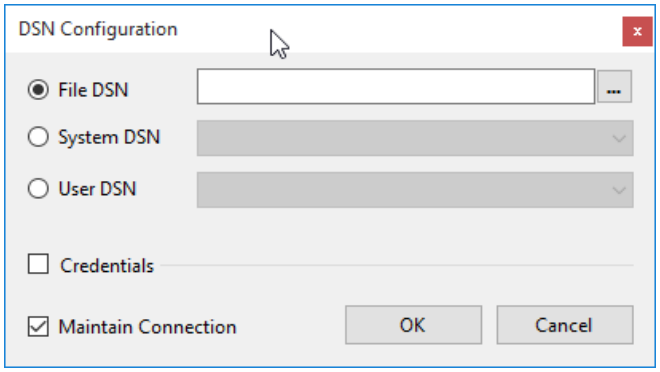


- **Access File** - Defines which MDB or ACCDB file will be used as the data source.
- **Show System Tables** – If checked, the Microsoft Access system tables will be included in the list of available tables.
- **Maintain Connection** – Will preserve the connection to the Access file, ensuring better performance but possible resulting in concurrency or file locking issues. By default, this property may be enabled.

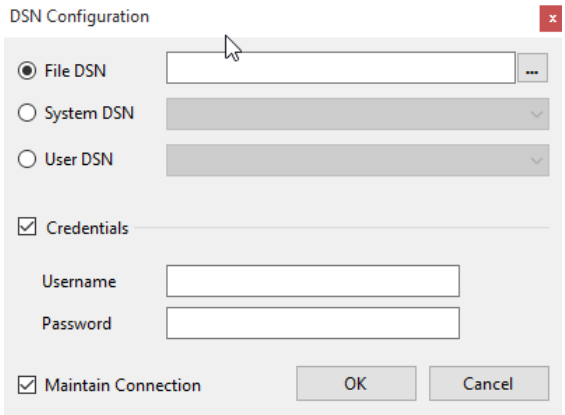
Note – The Access data source requires either Microsoft Office or, the freely available, Microsoft Access Database Engine. Without either installed, this data source will be unable to read text files.

DSN

This data source allows the Data Object to connect to a DSN. DSNs typically define a connection string to an arbitrary data source, limited by the drivers currently installed in the Windows ODBC subsystem. Once selected, the following dialog will appear with configuration options relevant to the Database data source.



- There are three possible types of DSNs supported, and only one may be selected by an individual Data Object at a time. DSNs may be created by using the ODBC Data Sources (64-bit) control panel widget in Windows.
    - **File DSN** – References a DSN file on the local file system.
    - **System FSN** – References a DSN stored in the Windows registry, accessible by any process and users of the system.
    - **User DSN** – References a DSN stored in the Windows registry, accessible by the currently logged in Windows user account.

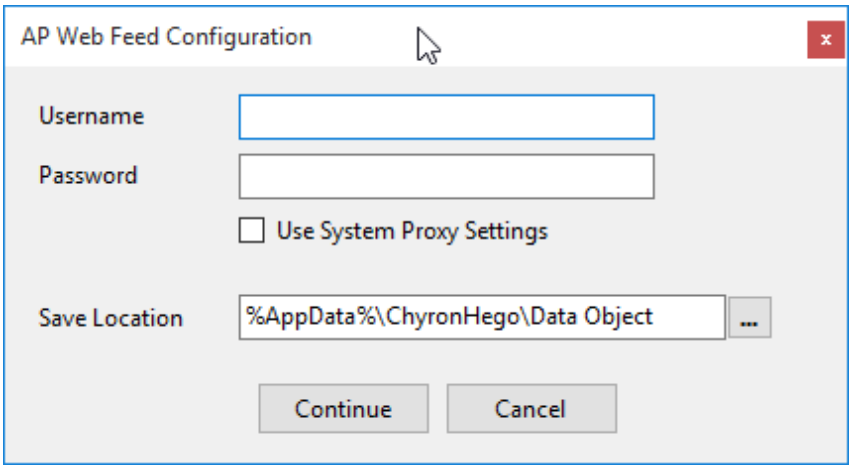


- **Credentials** – DSNs do not persist user credentials, but they may be specified by enabling the *Credentials* option.

- **Maintain Connection** – Preserves the database connection, ensuring better performance, but possibly resulting in concurrency or file locking uses. By default, this property may be enabled.

AP Web Feed

The AP Web Feed data source allows the Data Object to connect to the AP Web service using an existing account. Once selected, the following dialog will appear with configuration options relevant to the AP Web Feed data source.

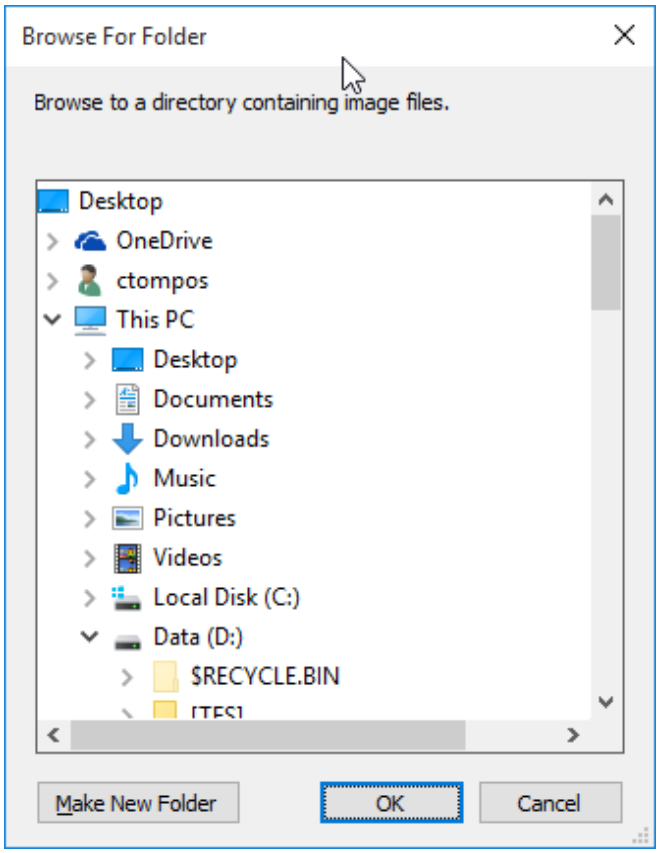


- **Username/Properties** – Used as credentials for the AP Web service
- **Use System Proxy Settings** – If enabled, the proxy settings configured under the Windows Internet Options widget will be observed.
- **Save Location** – Web data is downloaded regularly and stored on the local file system in the *Save Location*.

Images

The Images data source allows the Data Object to treat a local directory as a data source. A single data set is provided, including a list of all image files in the specified directory. This file list does not include images in subdirectories.

This functionality can be useful for cycling through images on an interval.



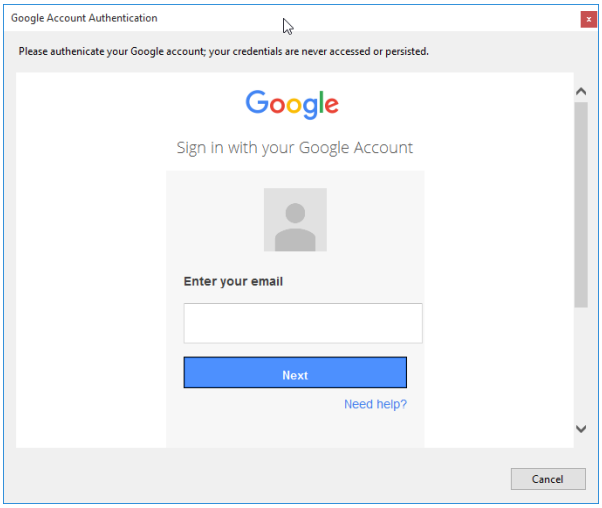
Movies

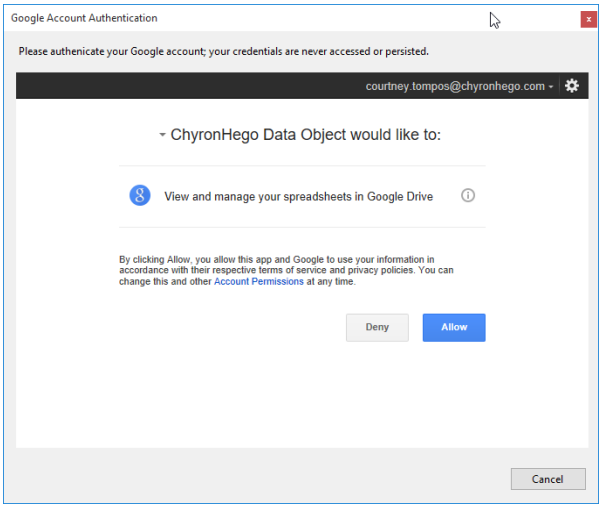
The Movies data source allows the Data Object to treat a local directory as a data source. A single data set is provided, including a list of all movie files in the specified directory. This file list does not include movies in subdirectories.

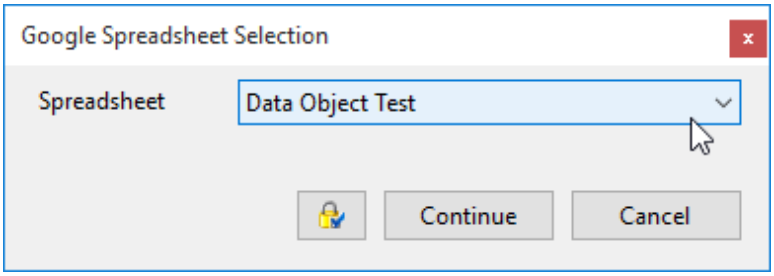
This functionality can be useful for cycling through movies on an interval.

Google

The Google data source allows the Data Object to connect to a Google spreadsheet. Initially, the user will have to specify a Google account and authorize access for the Data Object.

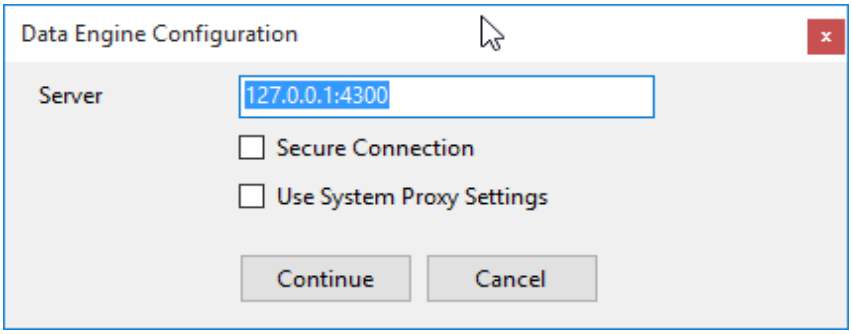






ChyronHego Data Engine

The Data Engine data source allows the Data Object to connect to a ChyronHego Data Engine bucket using Web Sockets.



- **Server** – Defines the address of the Data Engine service.
- **Secure Connection** – Specifies whether HTTPS should be used when connecting.
- **Use System Proxy Settings** – If enabled, the proxy settings configured under the Windows Internet Options widget will be observed.

Parameters

For backwards compatibility, the Data Object may target the Project and Scene parameter lists as a data set, presented in the form of a single record (Parameter name -> Column name). This binding strategy is no longer necessary, however, because PRIME now allows directly binding object properties to parameters. Previously, the Data Object was a required broker.

Filtering

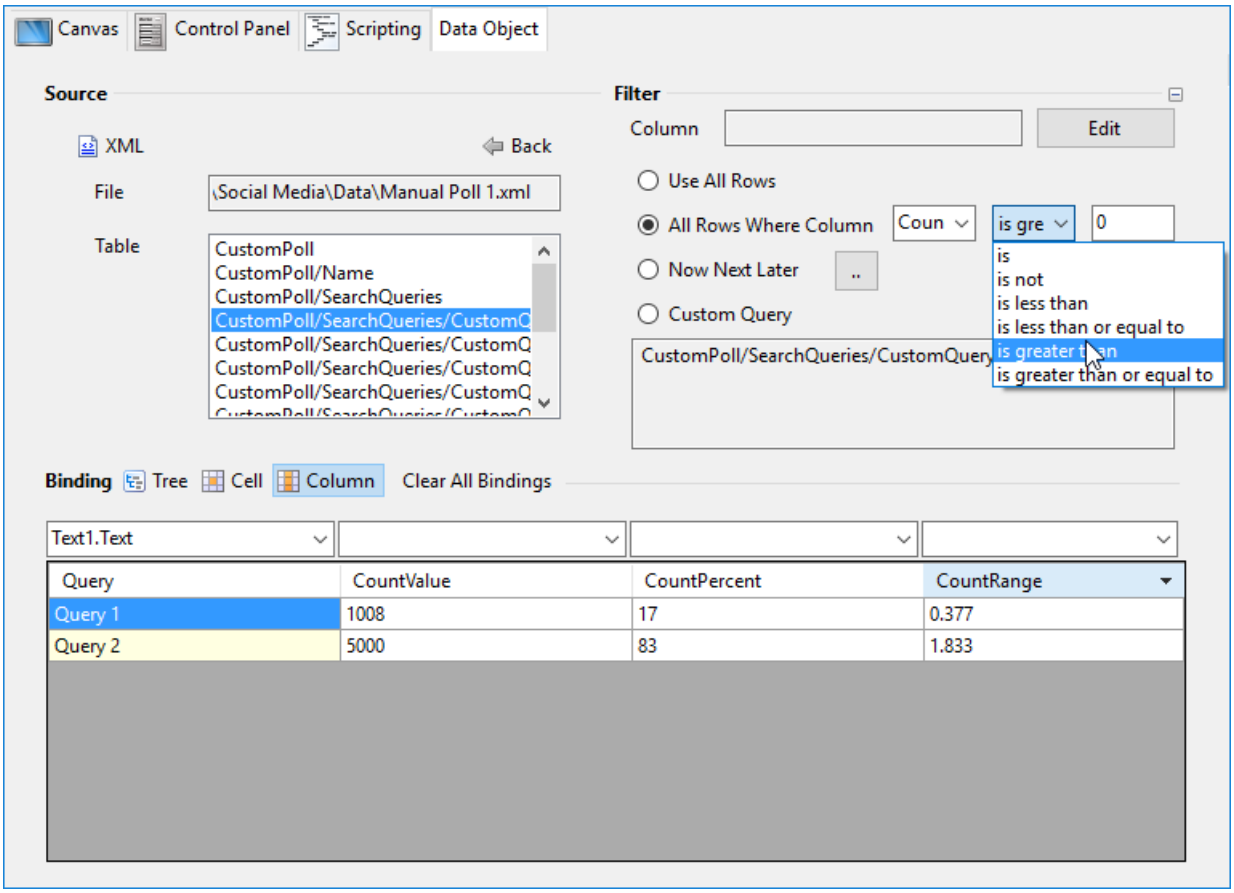
Once a specific data source has been selected, the user may choose one of the tables from the list to identify as a particular data set. Subsequently, additional filtering may be applied using one of the filters supported by the selected data source.

All Rows

This filter returns the entire data set. As no actual filtering is taking place, this is supported by all data source types.

All Rows Where Column

This filter returns the data set after restricting data based on a particular column. As a user selects a column, comparison operator and value, the visible data set will change dynamically. The query text will also change to reflect the updated filter.

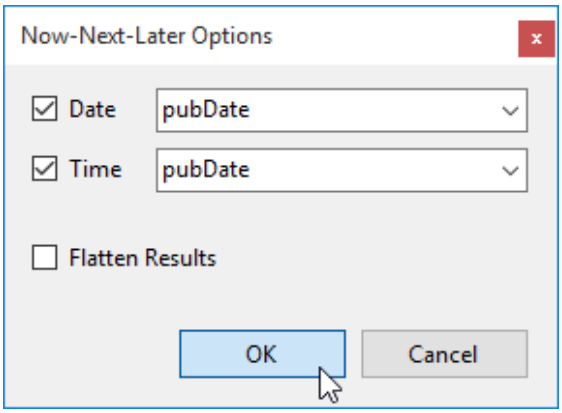


Note – This filter is **NOT** supported by the **JSON, Images, Movies** or **Parameters** data source types.

Now Next Later

This filter attempts to restrict the data set to a sequence of three records for use in a Now-Next-Later scenario. In such a scenario, the user may want the current record (presenting data closest to the current date/time), the next record (the next closest), and the record that follows.

These records are not necessarily adjacent to one another in the data set. However, they are ordered based on the Date/Time values contained in the targeted fields.



Note – This filter is **NOT** supported by the **Data Engine, Google, JSON, Images, Movies** or **Parameters** data source types.

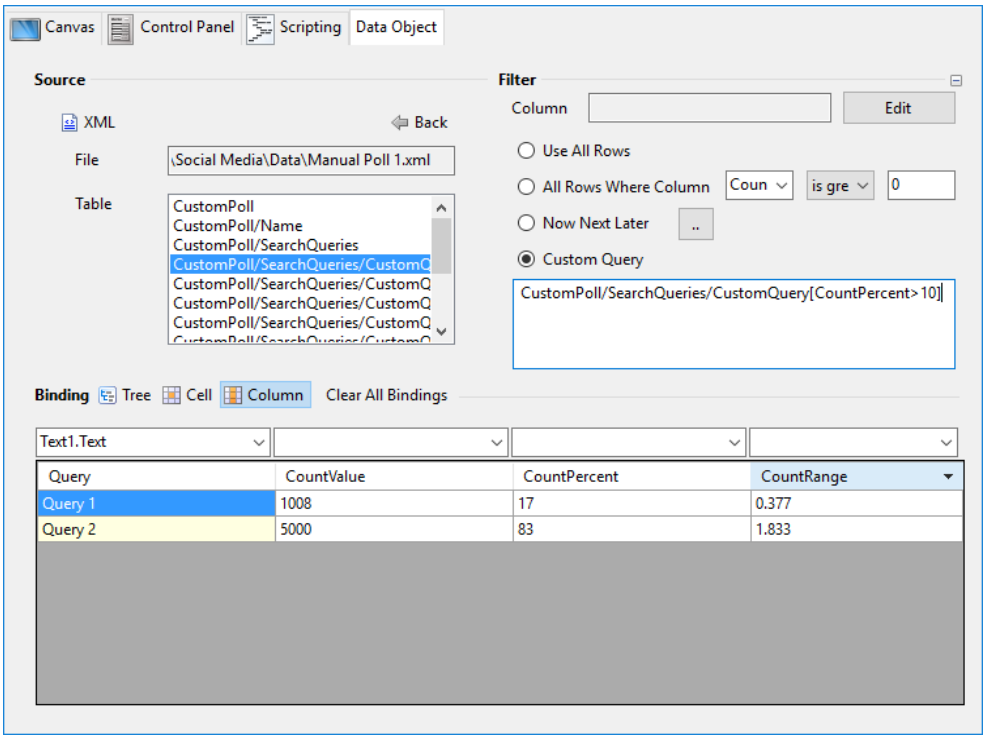
Custom Query

This filter restricts the data set based on a custom text query as defined by the user.

- In the case of the *Access*, *Excel*, *Text* and *Database* data source types, this query must take the form of a valid SQL selection statement.
- In the case of *AP Web Feed*, *XML*, *RSS*, and *Web* data source types, this query must take the form of a valid XPath query.
- In the case of the *Google* data source type, this query must take the form of a valid google spreadsheet query.

This filter restricts the data set based on a custom text query as defined by the user.

- In the case of the **Access, Excel, Text** and **Database** data source types, the query must take the form of a valid SQL selection statement.
- In the case of **AP Web Feed, XML, RSS** and **Web** data source types, this query must take the form of a valid XPath query.
- In the case of the **Google** data source type, this query must take the form of a valid Google spreadsheet query.



This filter is **NOT** supported by the **Data Engine, JSON, Images, Movies** or **Parameters** data source types.

Parameters in Query Strings

Parameters can currently be used in 4 places:

- the custom query text box
- the column name drop down (Match Column filter)
- the value text box (Match Column filter)
- the URL for RSS, Web and JSON providers

There are two parameter lists: **Project** and **Scene**

The syntax is the following:

{parameterListName::parameterName} or **{parameterListName.parameterName}**

Where **parameterListName** is either “Scene” or “Project”

Examples;

{Scene::Parameter 1}

{Scene::Player 1}

Google Sheet Query Example: **player = {Player 1}**

XML XPath Query Example: **data-set/record/{AwayScore}**

For Scene parameters you can avoid using the parameterListName and just apply the parameter name itself and it'll default to the Scene parameter list.

Example: **{Parameter 1}**

To access the projects parameters list use the “Projects” keyword.

{Project::Parameter 1}

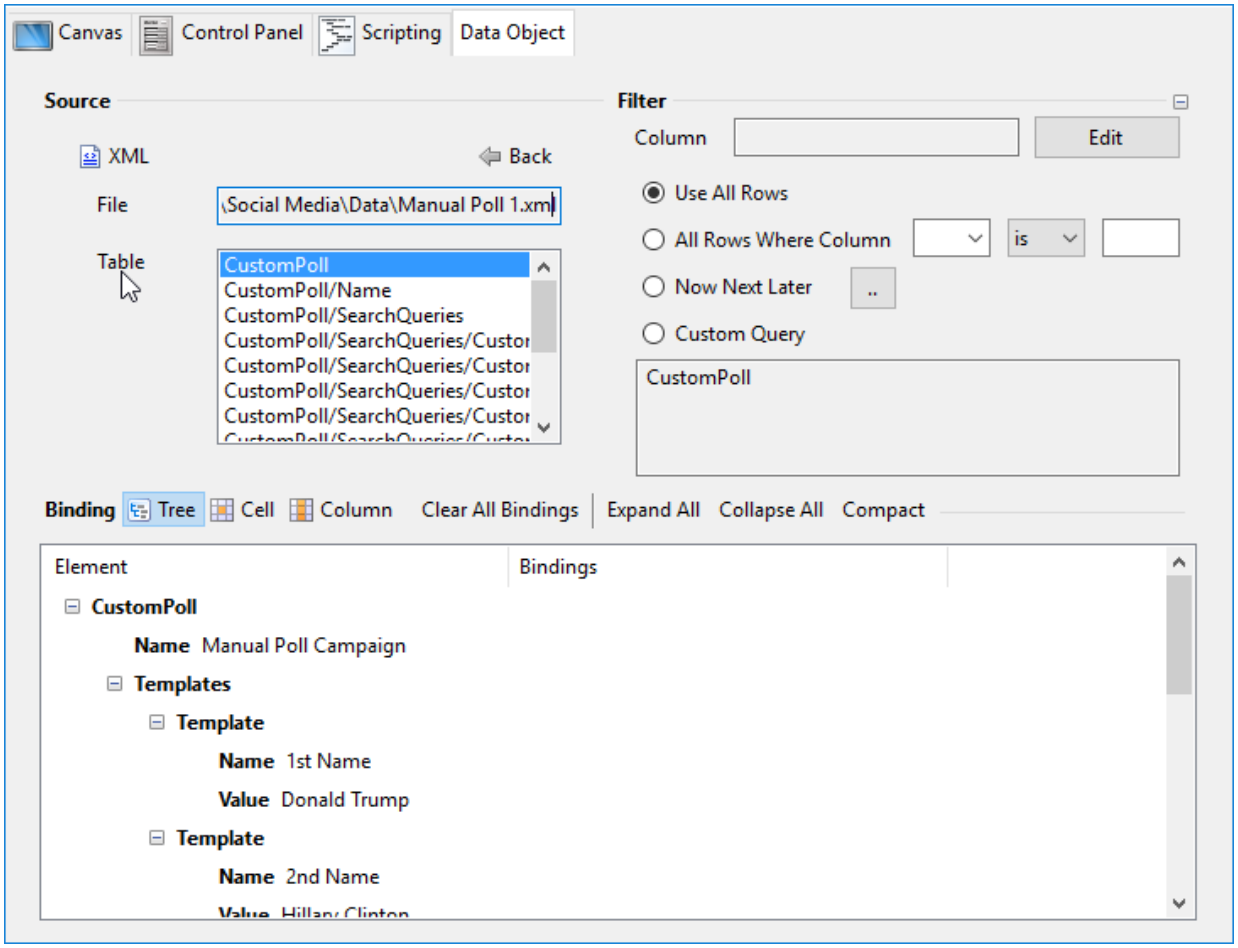
Binding Types

Once a data source and filter have been configured, the next step is choosing a binding type. Three binding models are available, although they are not universally supported across all data sources.

Tree Binding

Tree Binding is useful for interacting with a data model that is best expressed hierarchically. The data set is presented as a hierarchy of collapsible nodes, and the user may bind directly to any nodes with values.

Binding is addressed to a specific node within the hierarchy of the tree.

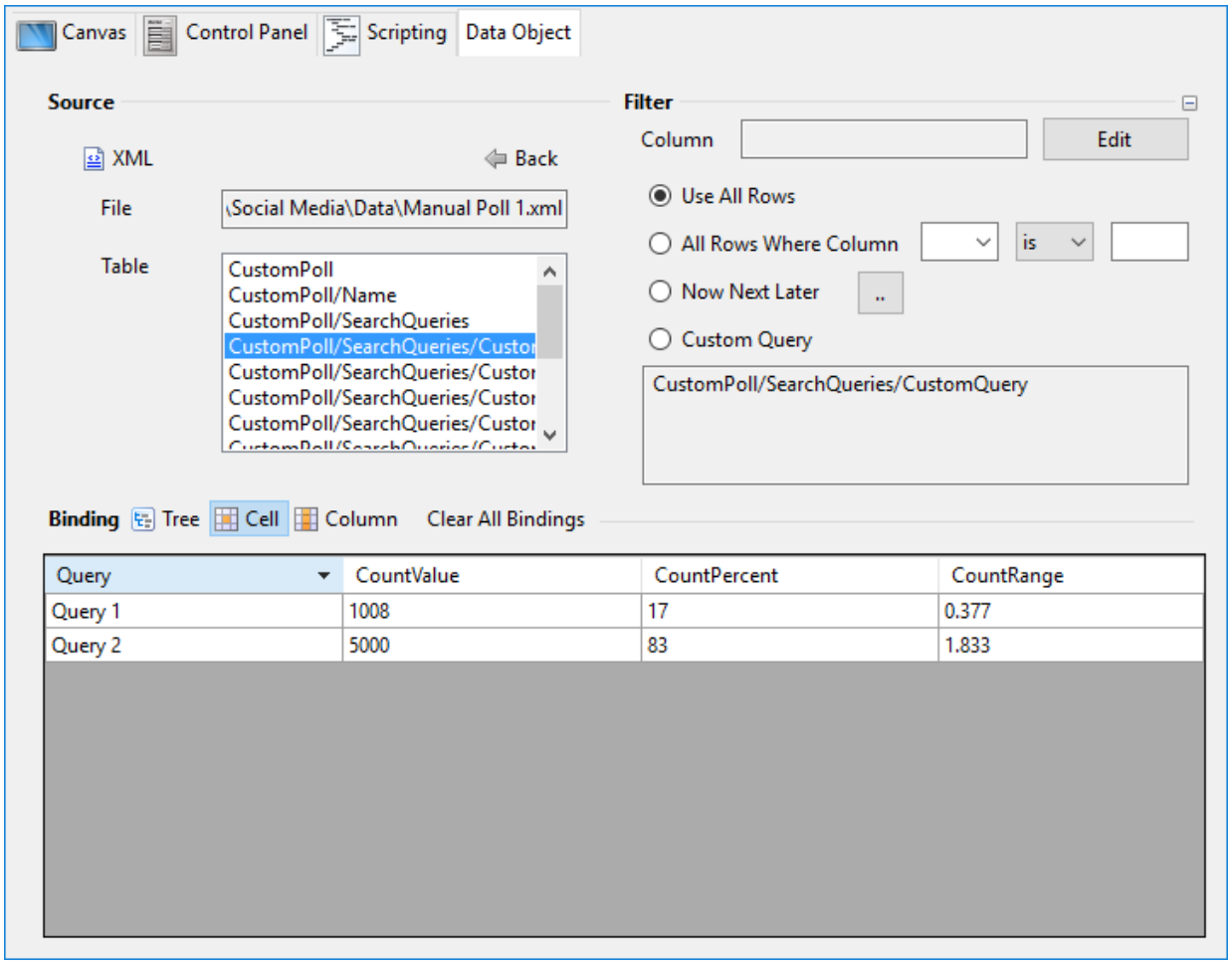


Tree Binding is supported by the **AP Web Feed, JSON, RSS, Web** and **XML** data source types.

Cell Binding

Cell Binding is useful for interacting with data that is best expressed in a tabular format. The data set is expressed as a series of rows containing one or more columns.

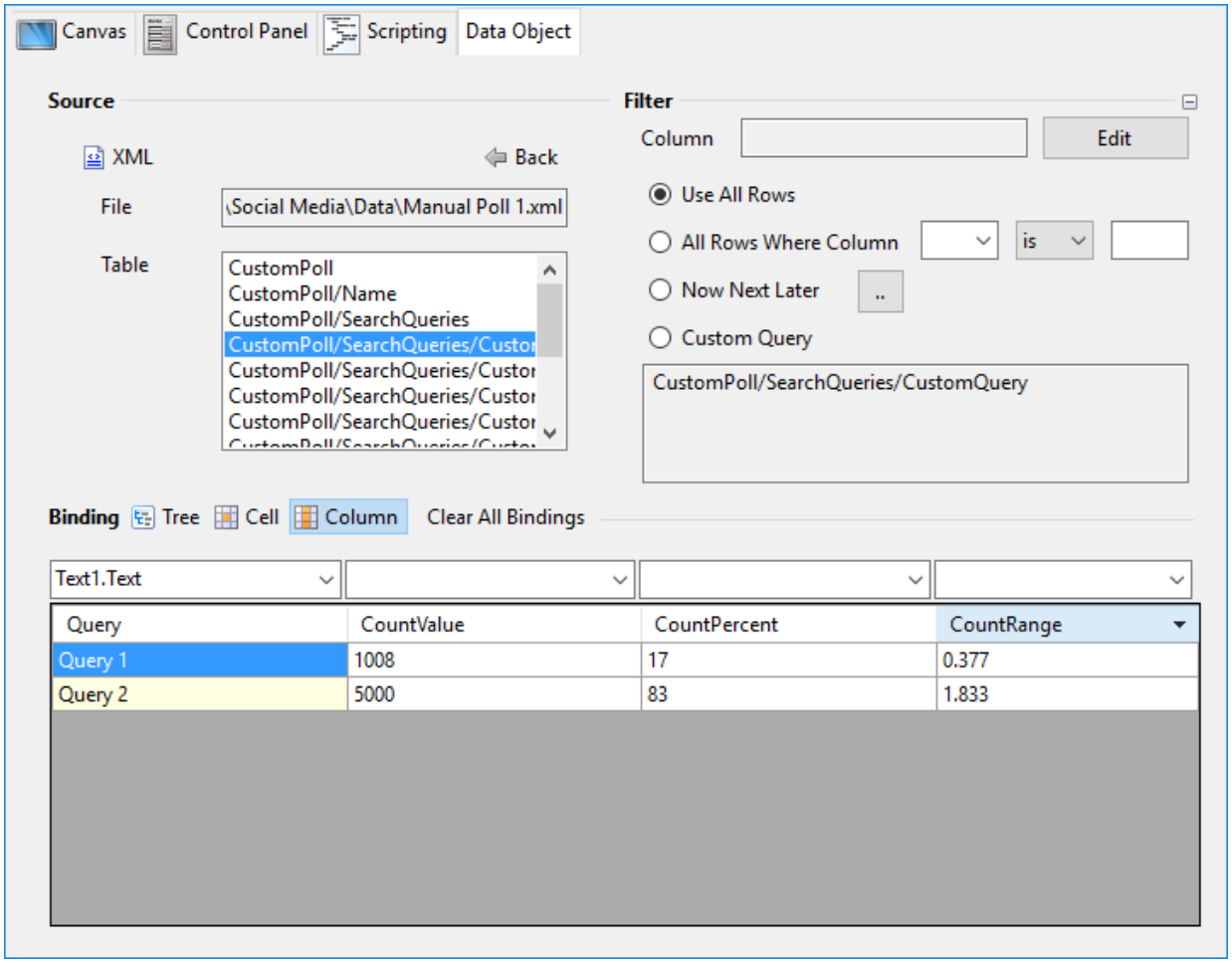
Binding is addressed to a specific field (column name, row number) within the data set.



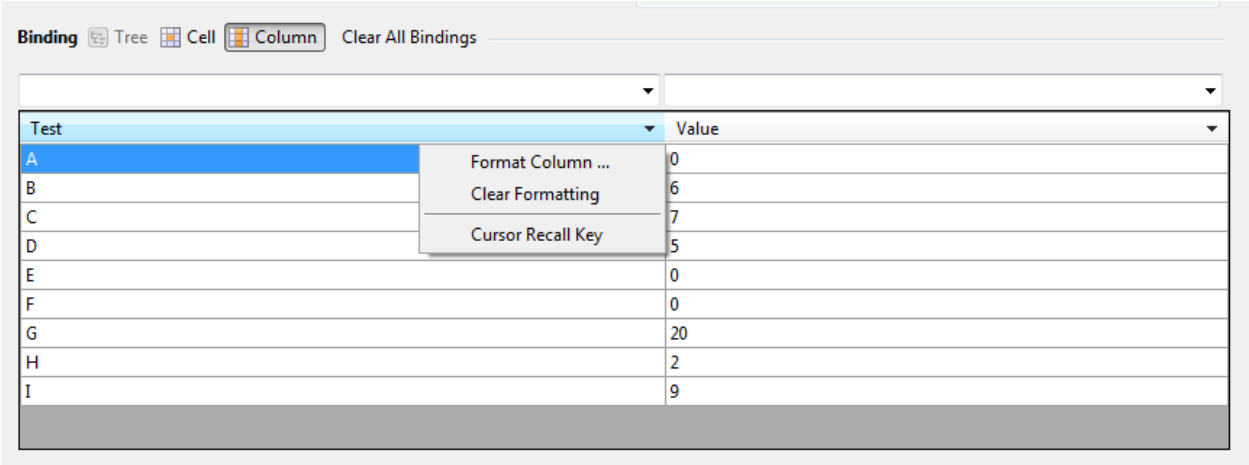
Cell Binding is supported by all data source types.

Column Binding

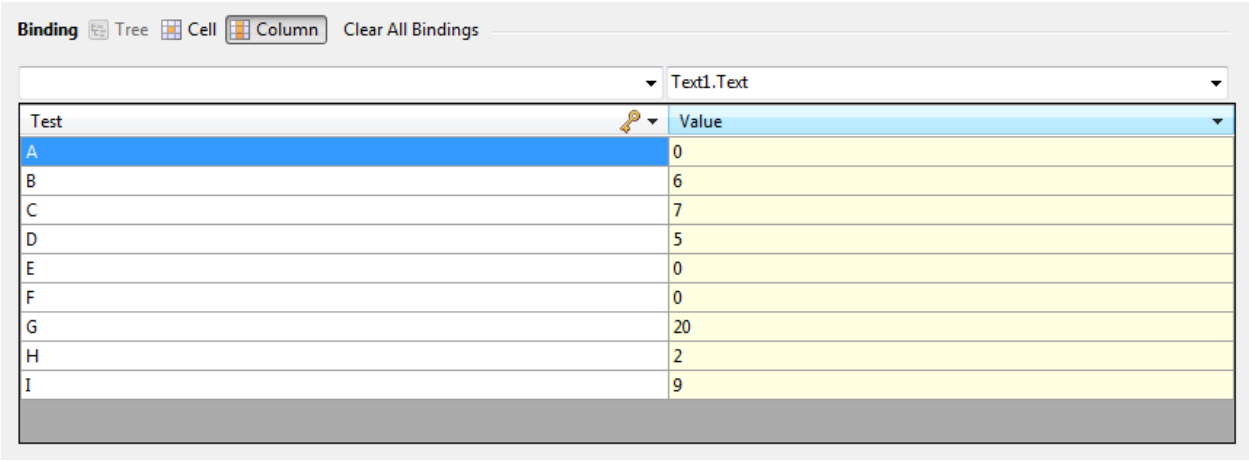
Similar to Cell Binding, Column Binding is also useful for interacting with data that is best expressed in a tabular format. However, Column Binding is instead addressed to an entire column with the assumption that the Data Object will navigate through each data record using commands. To the contrary, Cell Binding targets a specific record and navigation has no effect.



Column binding also offers the ability to automatically navigate back to the current record if the backing data has changed. This is useful, for example, if the data source is updated and additional data has been added before the record currently being displayed or if data has been deleted. To utilize this functionality, right-click on a column known to contain unique data and check the Cursor Recall Key option.



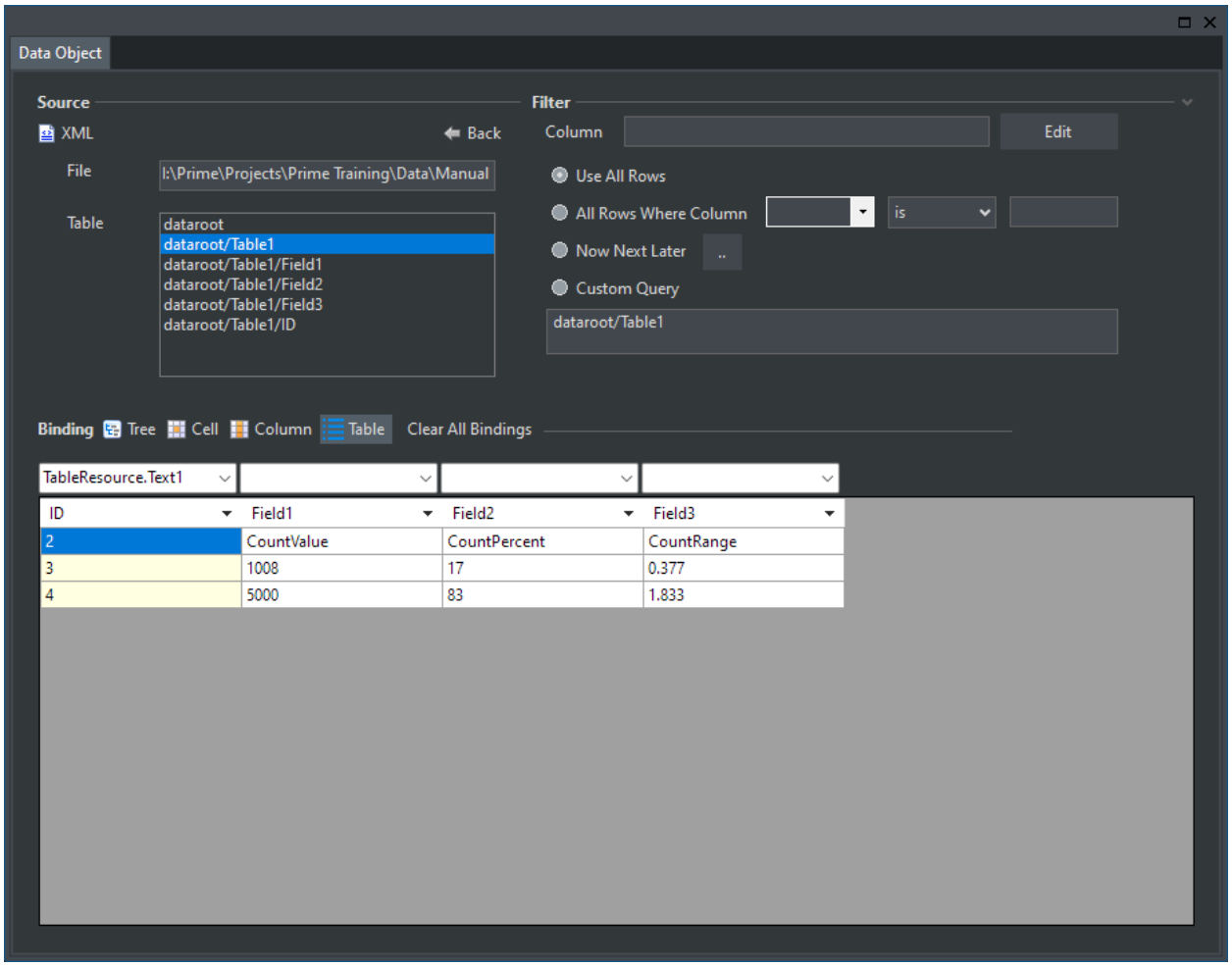
Once enabled, a key icon will appear on the column header. Data from this column will now be used to determine where the cursor is as we navigate through the data set. The column specified as the key does not have to be data bound; the only requirement for this functionality to work properly is that all of the data in this column must be unique. Consequently, this is most useful when combined with the database index or primary key concept, however any unique column will suffice.



The Cursor Recall Key feature is only available in column binding mode.

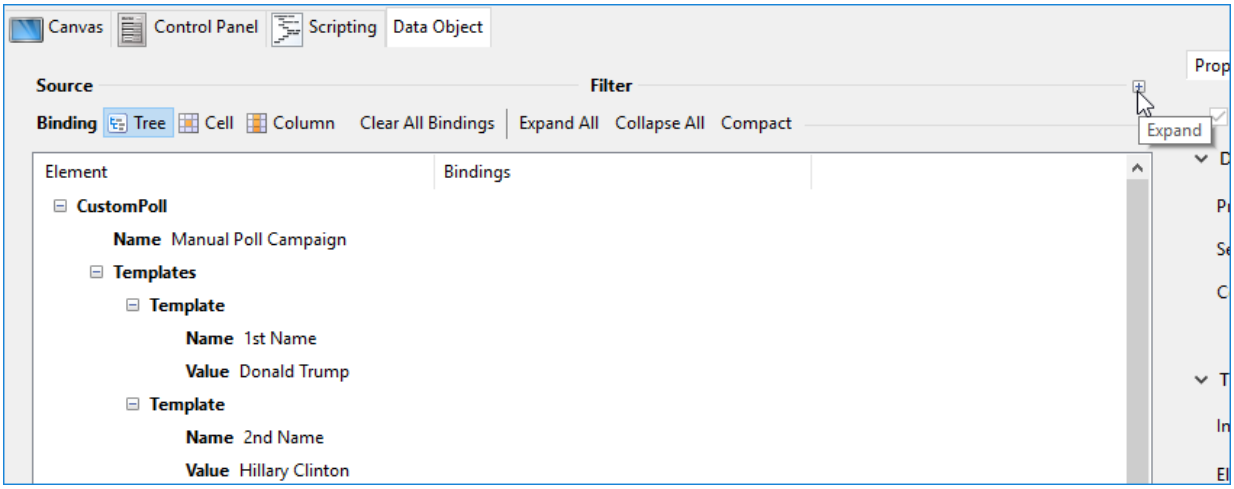
Table Binding

Table Binding allows users to populate the data in Table Resource using Data Object sources. Similar to Column Binding, they can bind the tabular data columns to columns defined by Table Resource.



How to Bind Targets

To facilitate binding, the Source and Filter sections can be resized manually, or hidden completely by clicking the Expand/Collapse caret as seen below:

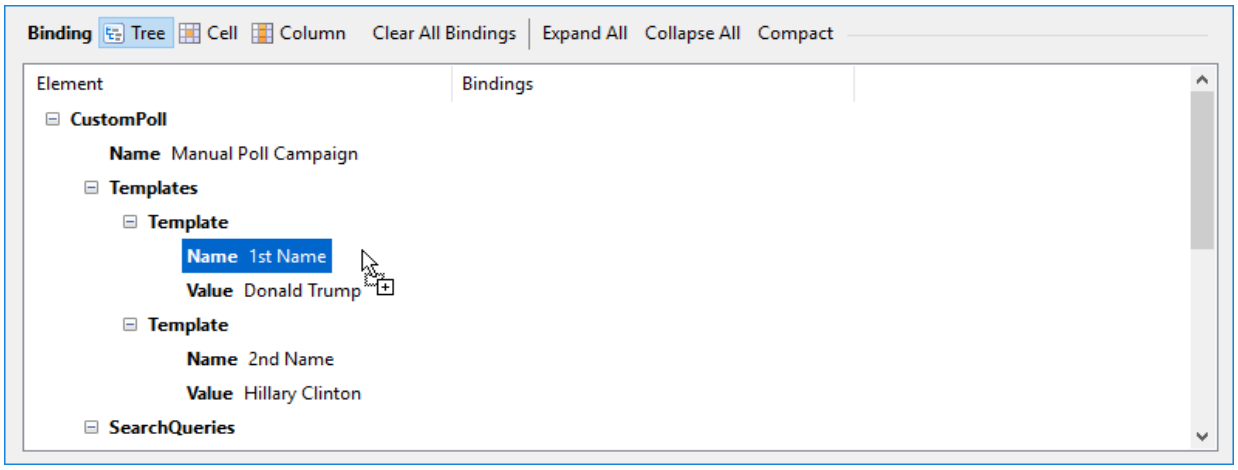


Drag and Drop

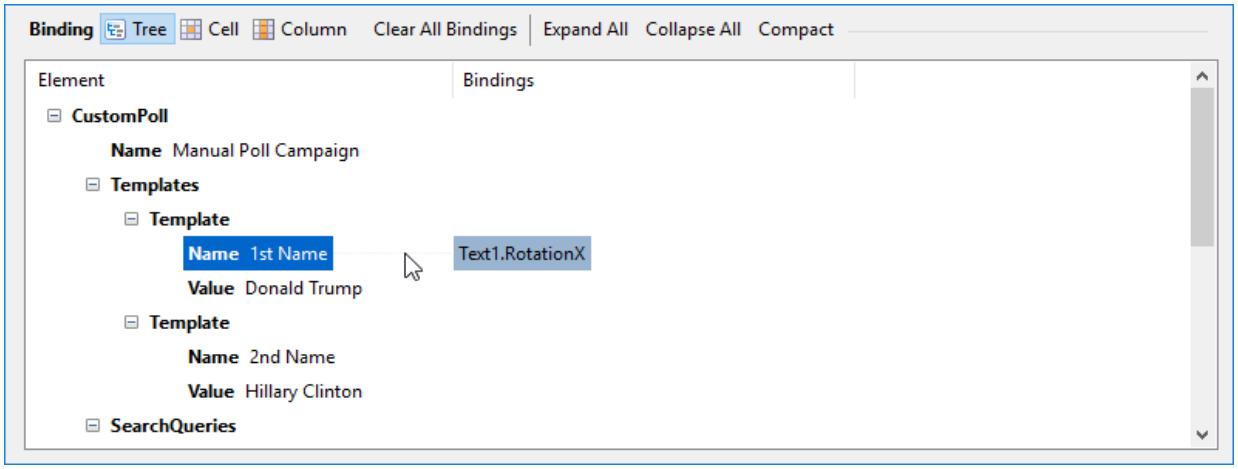
Binding may be accomplished using drag-and-drop by dragging properties from one of three possible locations and dropping them onto the binding section of the Data Object Panel.

- Dragging objects from the Scene Tree will result in binding the Default property of the object being dragged. For example, dragging a Text object will result in binding the Text property (e.g. Text1.Text). Dragging and Image object will result in binding the File property (e.g. Image1.File).
- Dragging properties from the Property panel will result in binding the object (e.g. Text1.Text or Image1.File).
- Dragging properties from the Keyframes panel will result in binding the keyframe (e.g. Text1.Action1.Keyframe1.Opacity).

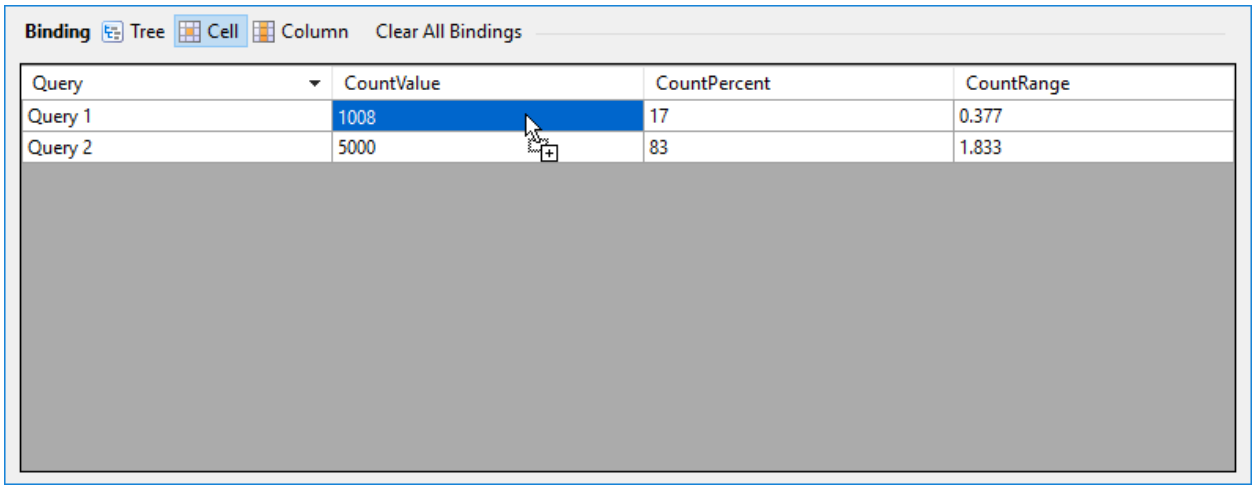
In Tree Binding, drag an item onto a desired node.



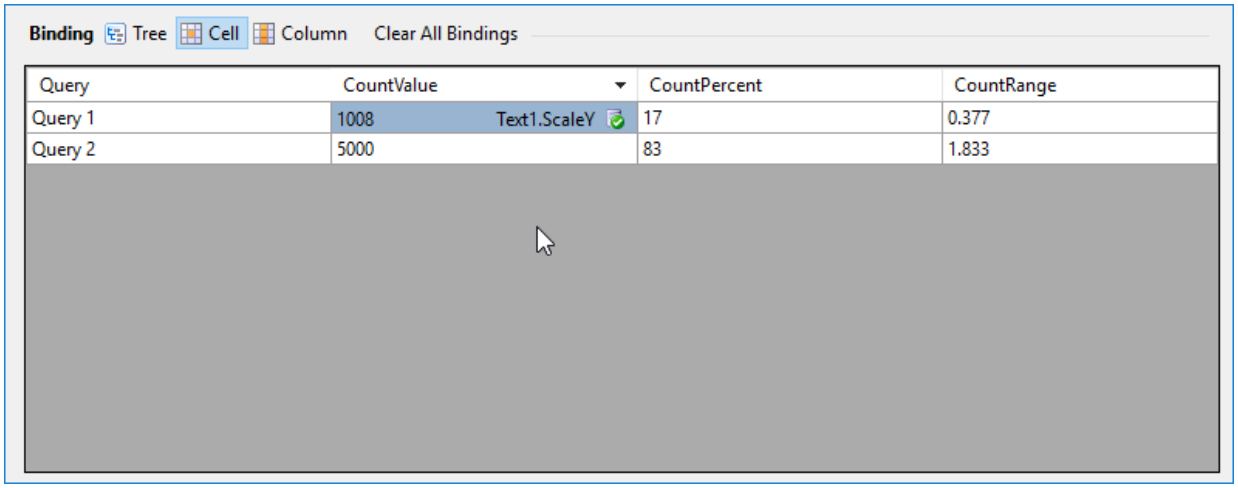
Dropping the item creates the binding.



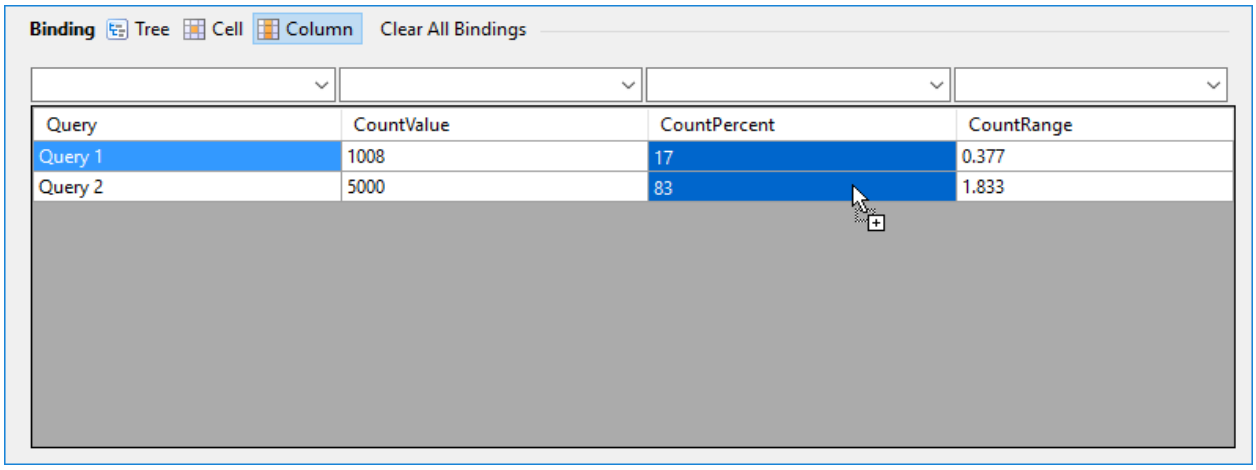
In Cell Binding, drag an item onto a desired cell/field.



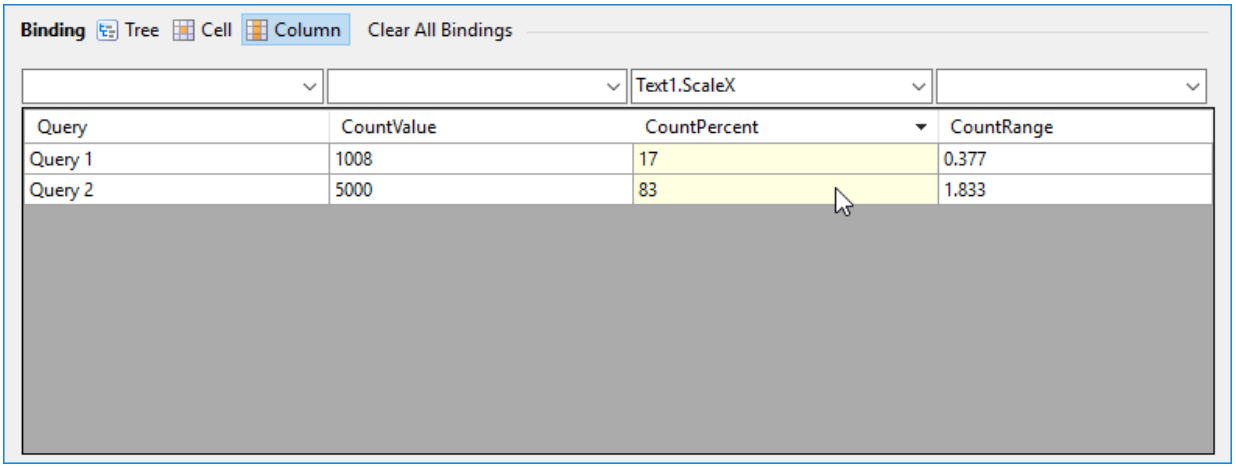
Dropping the item creates the binding.



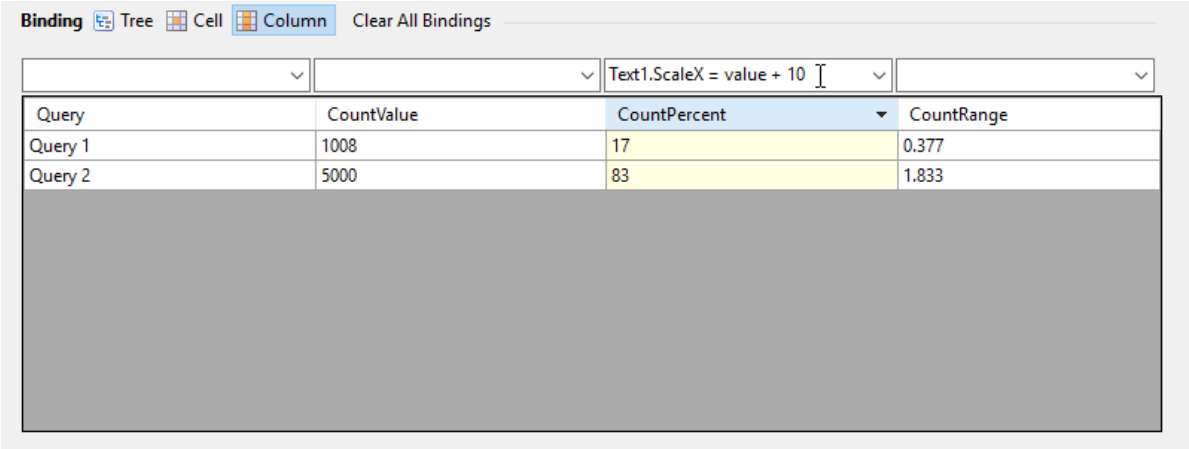
In Column Binding, drag an item onto the desired column.



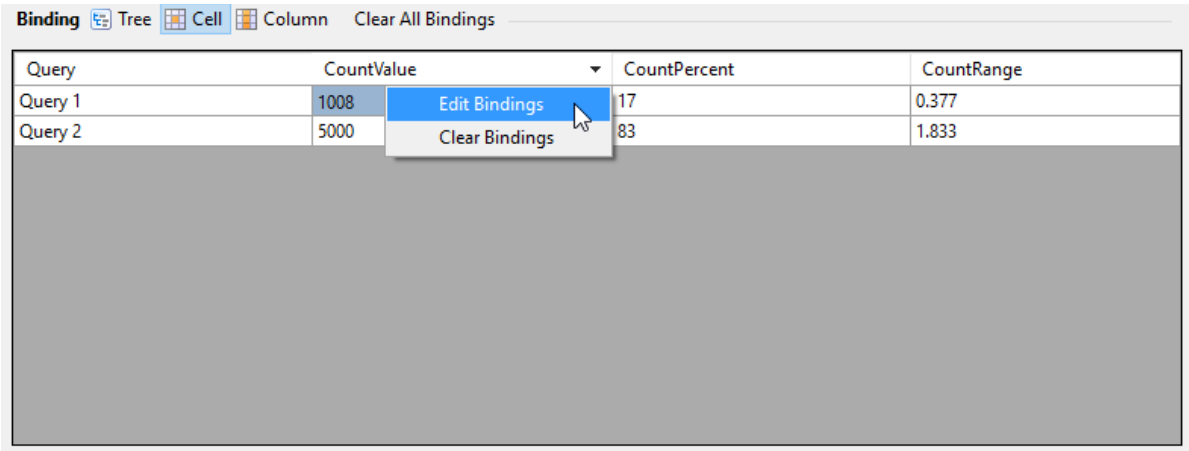
Dropping the item creates the binding.



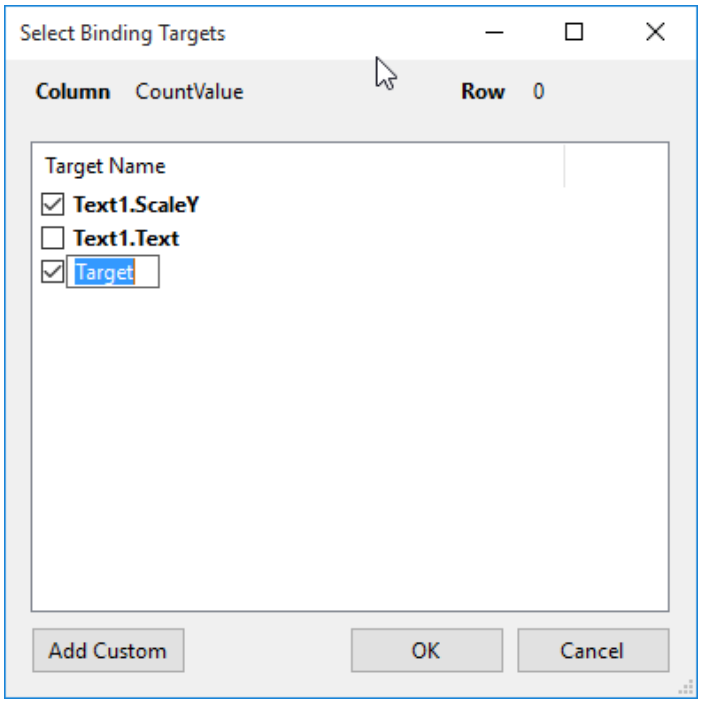
Bindings can be simple, as created by drag-and-drop, or they can use the same syntax and expressions supported by bindings in other PRIME features. For example, the user may type an expression in the column selector in Column Binding mode.

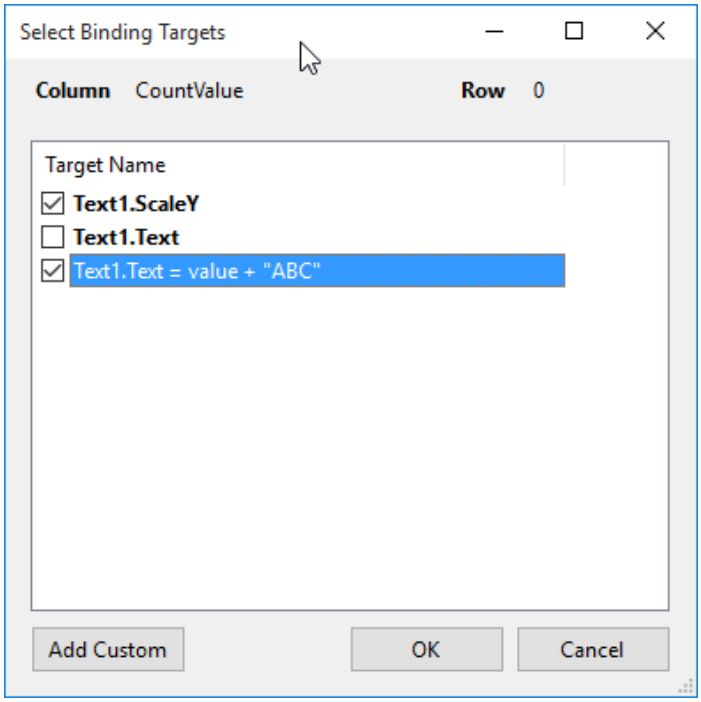


Alternatively, the user may right-click and choose Edit Bindings for a more advanced binding editor.



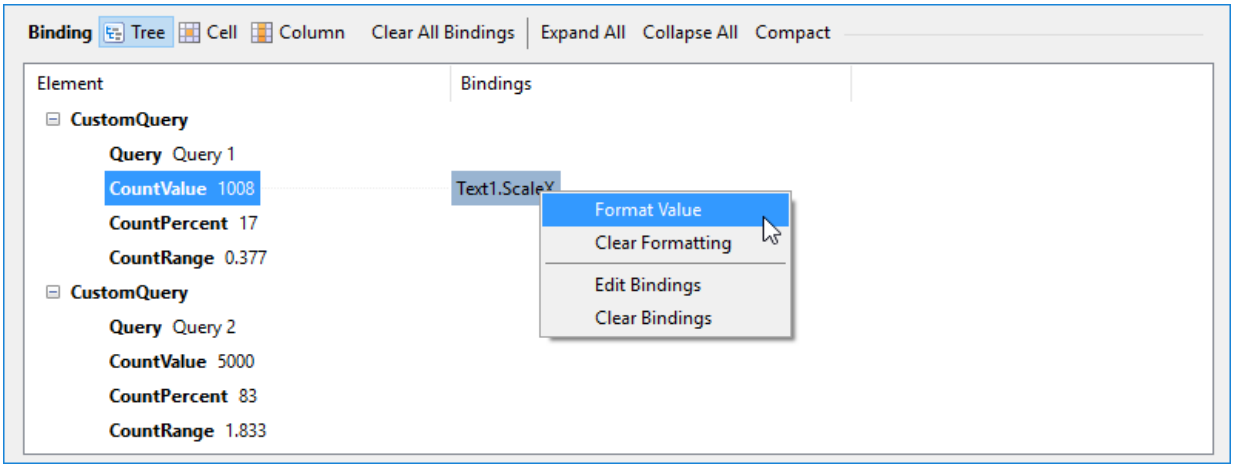
From this dialog, the user may add or remove bindings from the default list, or add custom binding expressions using the “Add Custom” button.





Formatting Data

All binding modes support custom data formatting. In Tree Binding, the user may right-click an individual node and choose the “Format Value” option from the context menu.



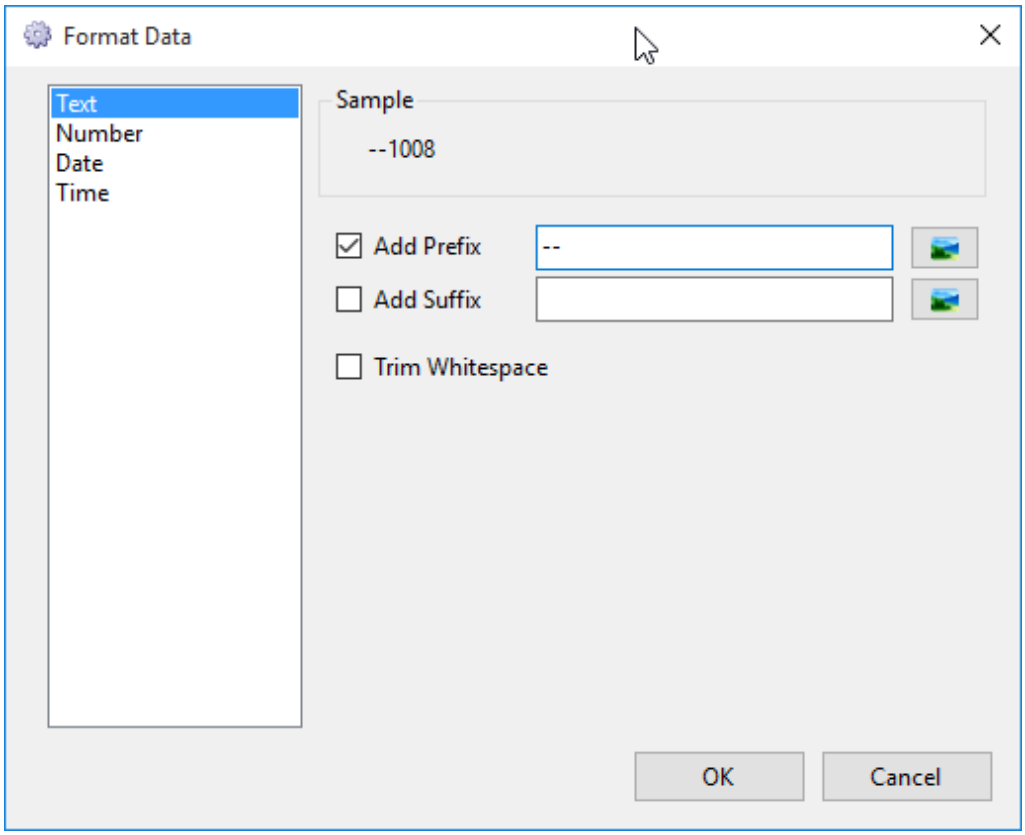
Similarly, in Cell Binding the user may right-click an individual cell and choose the “Format Value” option from the context menu.

Data may be configured as either **Text, Number, Date** or **Time**. If the current data value cannot be formatted in the desired format, the existing value will remain intact and unaltered.

The user may select the desired format and then configure appropriate options.

Text

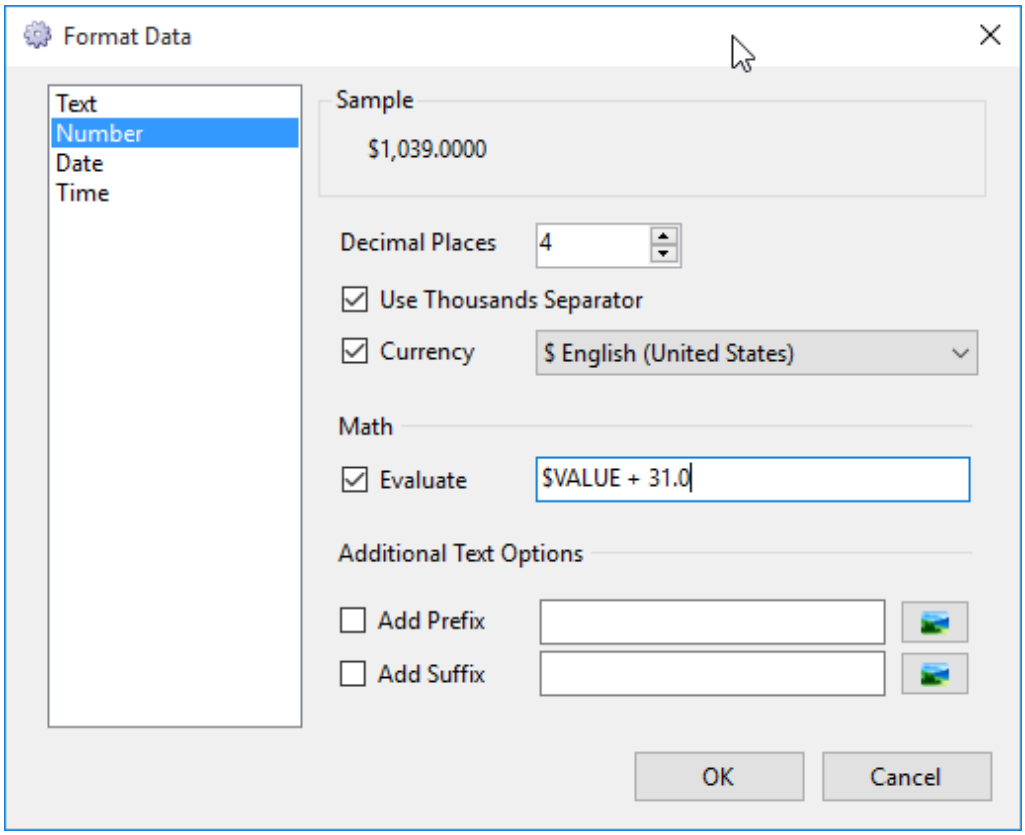
This data format will treat field data as plain text. This is the default format.



- **Add Prefix** – Adds text before the current value.
- **Add Suffix** – Adds text after the current value.
- **Trim Whitespace** – Removes leading and trailing non-printable characters.

Number

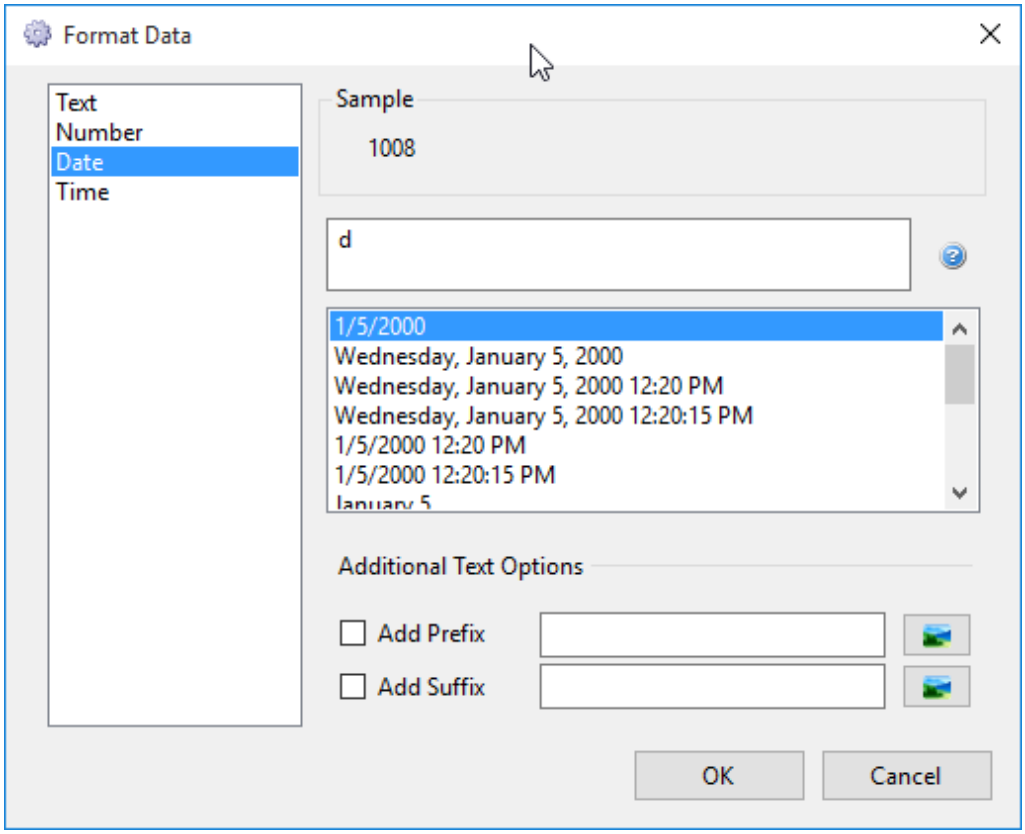
This data format will attempt to treat field data as numeric values. If a given data field is non-numeric, it will be unaffected by this formatting.



- **Decimal Places** – Specifies the number of digits that will appear after the decimal. If set to 0, the number will appear as a rounded integer.
- **Use Thousand Separator** – Indicates whether a culture-specific separator will appear (e.g. 1,000 as opposed to 1000).
- **Currency** – Indicates whether the number should appear as a currency value. If enabled, the user may then choose whichever currency symbol is appropriate.
- **Evaluate** – Supports basic arithmetic support for modifying field data.
    - E.g. "$Value + 31.0" will add the number 31.0 to any data field formatted in this manner.
- **Add Prefix** – Adds text before the current value.
- **Add Suffix** – Adds text after the current value.

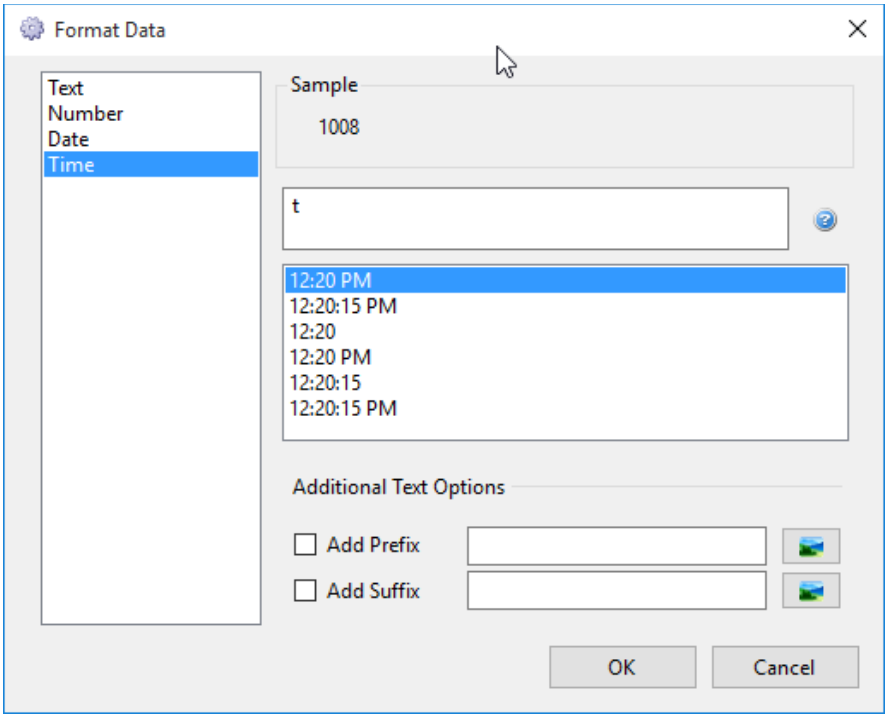
Date

This data format will appear to treat field data as date values. If a given data field is not a date value, it will be unaffected by this formatting. The user may apply one of the default data formatting strings, or create a custom one using the standard .NET DateTime formatting conventions.

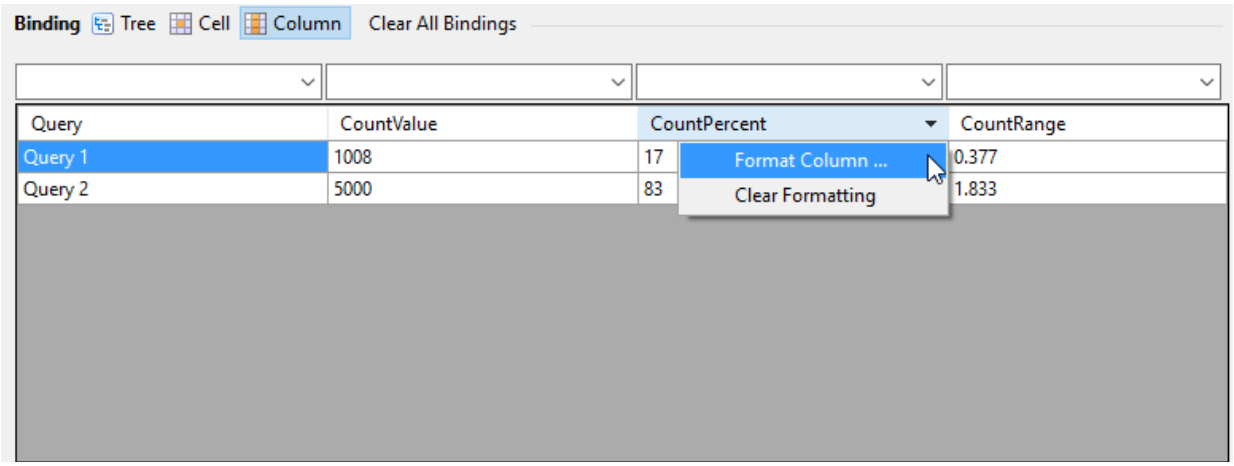


Time

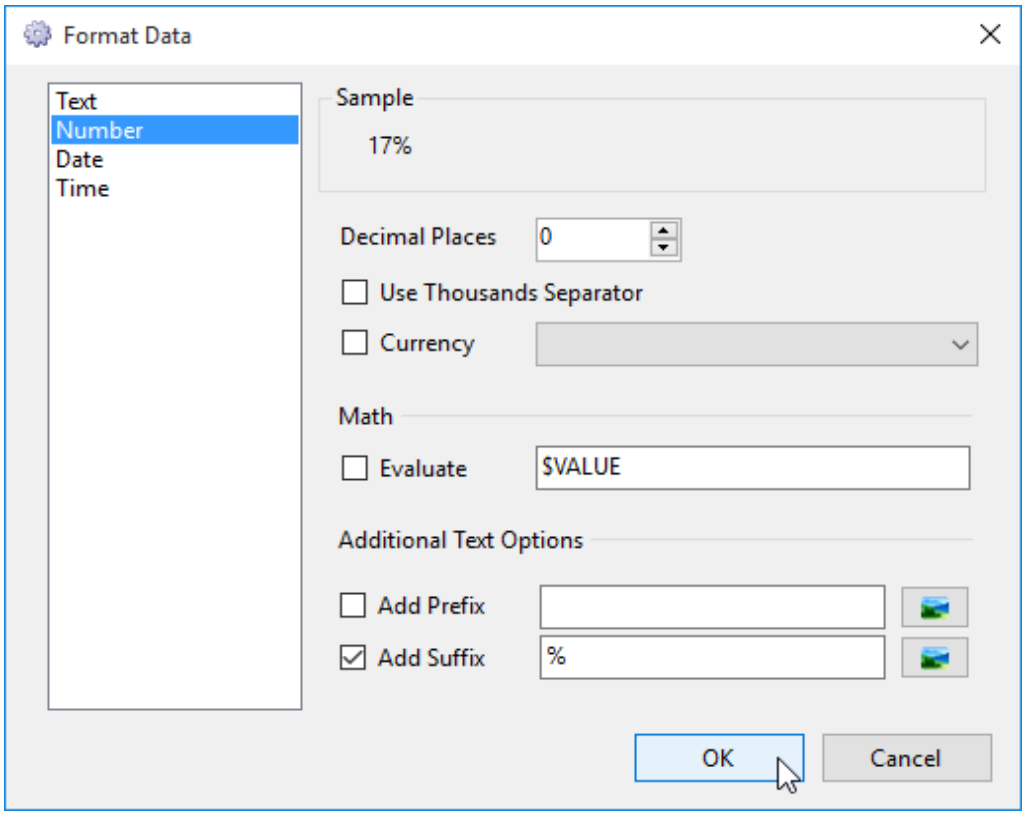
This data format will attempt to treat field data as time values. If a given data field is not a time value, it will be unaffected by this formatting. The user may apply one of the default data formatting strings provided or create a custom one using the standard .NET DateTime formatting conventions.



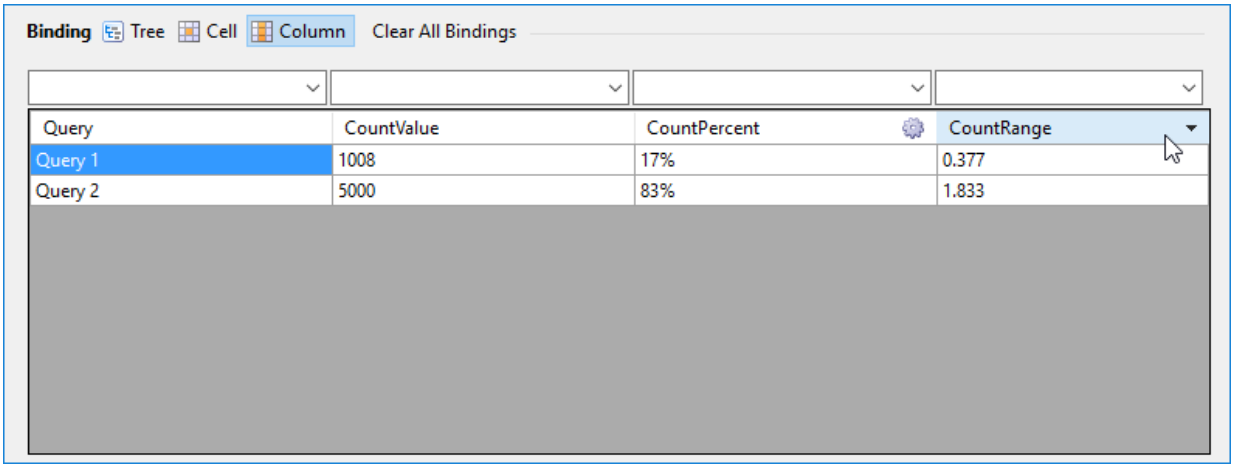
In Column Binding, the user may format an entire data column by right-clicking a column header as seen below:



For example, the user may want to format the “CountPercent” value seen above so that it includes a percentage symbol. One way to accomplish this is to choose the Number format type, and then add a custom Suffix with the percent symbol (%).



Once applied, the data preview displayed in the binding section is updated to reflect the inclusion of the percentage symbol, and a gear icon appears on the column header to indicate that formatting is enabled.


Binding
Tree
Cell
Column
Clear All Bindings
Query
CountValue
CountPercent
CountRange
Query 1
1008
17%
0.377
Query 2
5000
83%
1.833

Commands

Once a data source has been selected and binding has been configured, the next step is to determine how a PRIME scene interacts with the data source. At the most basic level, a Data Object may issue commands using the Command property.

In Tree and Cell Binding, bindings map specific data fields within the data set to targets within the scene. For example, Text1.Text may be bound to a cell in the second row of a specific column.

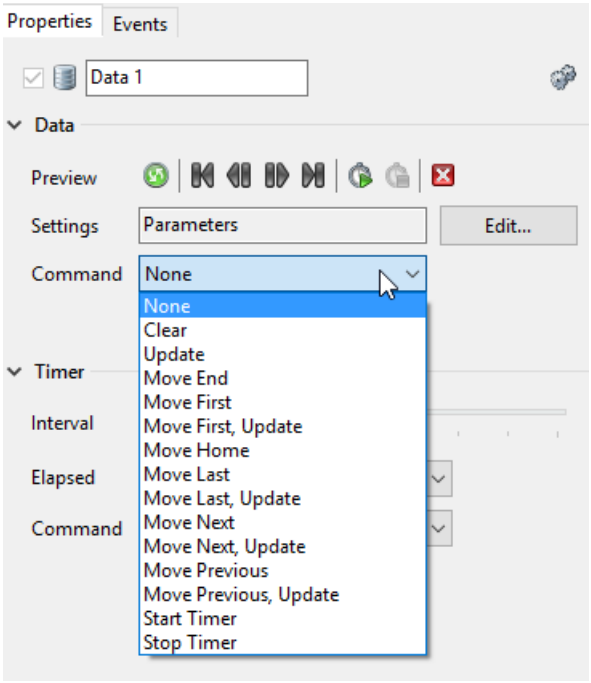
To the contrary, in Column Binding, bindings map entire columns with the expectation that the scene will navigate through each record using Data Object commands.

With Column Binding, the Data Object maintains an internal cursor that identifies the current location within the data set.

- BOF (Before the First Record)
- Record 1
- …
- Record N
- EOF (After the Last Record)

Because Tree and Cell Binding target specific fields, the notion of a cursor does not apply and the navigation commands have no effect.

The full list of supported commands, and their effects, is listed below:



Command	Result	Notes
None	Do nothing.	--
Clear	Clears all data bound targets. For example, if a Data Object has bound a field to Text1.Text, then the Clear command will update Text1.Text with an empty text string.	--
Update	Applies the current values	This command

	to bound targets.	will not affect the internal cursor when executed in Column Binding mode.
Move End	Moves the internal cursor to the position immediately following the last record in the data set (EOF).	This command has no effect in Cell and Tree Binding mode.
Move First	Moves the internal cursor to the first record in the data set.	This command has no effect in Cell and Tree Binding mode.
Move First, Update	Moves the internal cursor to the first record in the data set and then applies the values to bound targets.	The Move First portion of this command has no effect in Cell and Tree Binding mode, however the Update will be executed normally.

Move Home	Moves the internal cursor to the position immediately preceding the first record in the data set (BOF).	This command no effect in Cell and Tree Binding mode.
Move Last	Moves the internal cursor to the last record in the data set.	This command has no effect in Cell and Tree Binding mode.
Move Last, Update	Moves the internal cursor to the last record in the data set and then applies the values to bound targets.	The Move Last portion of this command has no effect in Cell and Tree Binding mode, however the Update will be executed normally.
Move Next	Moves the internal cursor to the next record in the data set.	This command has no effect in Cell and Tree Binding mode. If this results in EOF and looping is enabled on the Data

		Object, then the cursor will wrap around to the first record.
Move Next, Update	Moves the internal cursor to the next record in the data set and then applies the values to bound targets.	The Move Next portion of this command has no effect in Cell and Tree Binding mode, however the Update will be executed normally.
Move Previous	Moves the internal cursor to the previous record in the data set.	This command has no effect in Cell and Tree Binding mode. If this results in BOF and looping is enabled on the Data Object, then the cursor will wrap around to the last record.

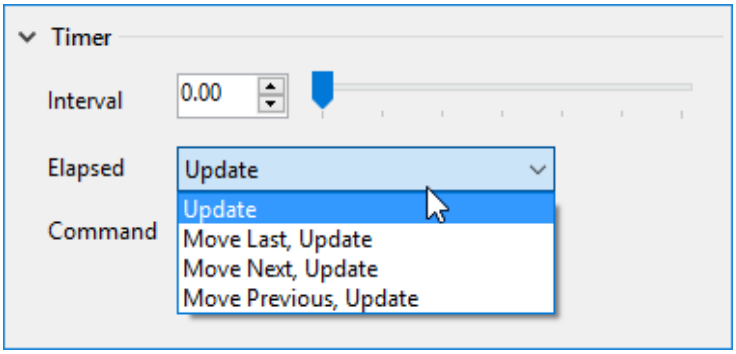
Move Previous, Update	Moves the internal cursor to the previous record in the data set and then applies the values to bound targets.	The Move Previous portion of this command has no effect in Cell and Tree Binding mode, however the Update will be executed normally.
Start Timer	Starts the Data Object timer for interval based command execution.	--
Stop Timer	Stops the Data Object timer for interval based command execution.	--

Timeline

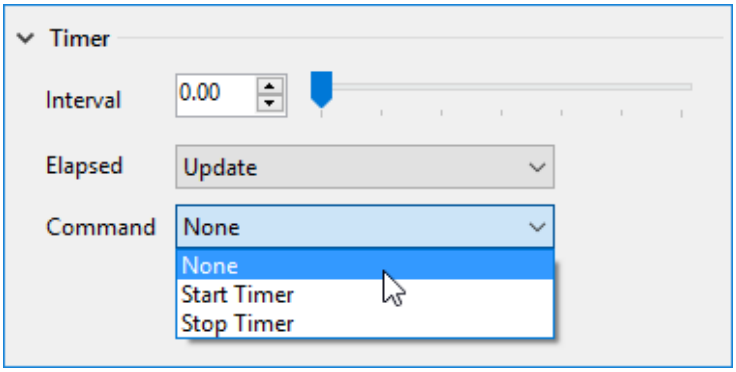
The Data Object command property is keyframeable; consequently, it is possible to manage data source navigation and updates entirely through the timeline.

Timer

Alternatively, the Data Object may be configured to execute a particular command on a set interval using the built-in timer functionality. By default, the **Timer Interval** property is set to 0.0 seconds. With this value, the Timer cannot be enabled; only non-zero intervals are valid.



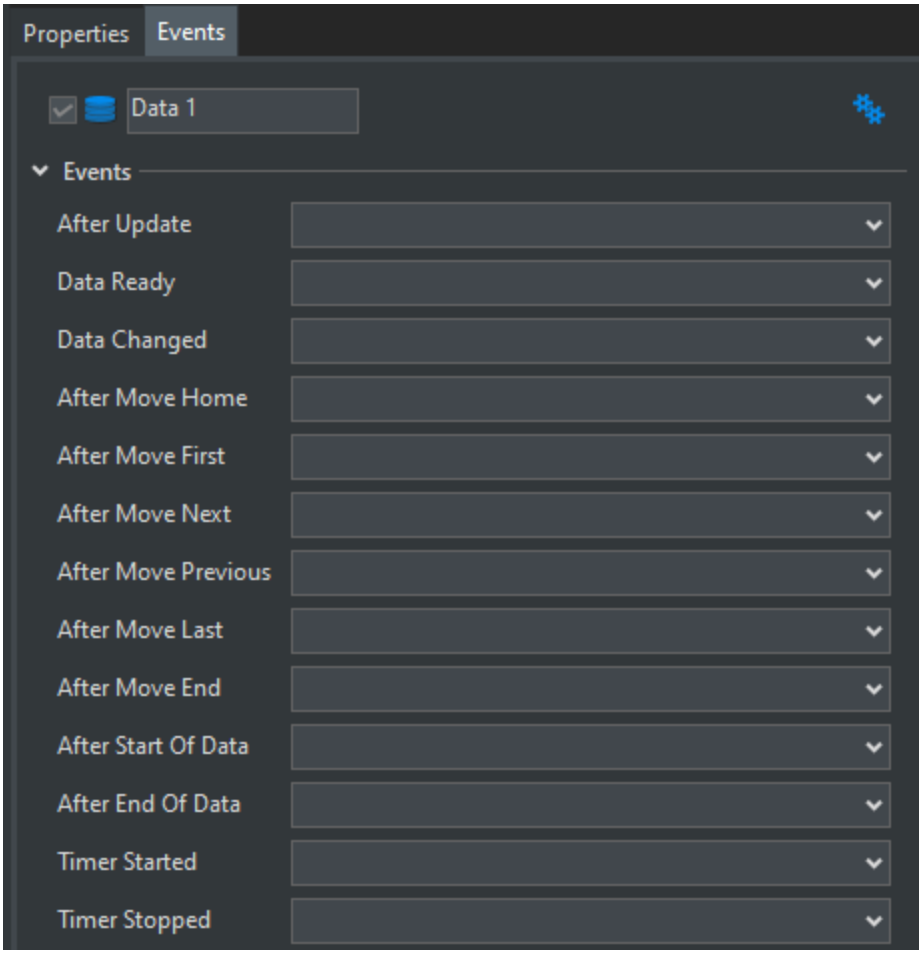
The **Timer Elapsed** property dictates the Data Object command that will execute whenever the interval elapses.



The **Timer Command** is a keyframeable property, which may be used to Start and Stop the timer.

Events

The Data Object supports events that may be used to trigger other aspects of the scene.



- **After Update** – Triggered whenever an Update command is executed. This pertains to the standalone Update command, as well as those coupled with navigation.
    - Update
    - Move First, Update
    - Move Last, Update
    - Move Next, Update
    - Move Previous, Update
- **Data Ready** - Triggered when data has been received and value is not empty.
- **Data Changed** - Triggered when new data is detected. Available as data source trigger for: Text, RSS, Web, JSON and Excel
- **After Move Home** – Triggered whenever a Move Home command is executed.
- **After Move First** – Triggered whenever a Move First command is executed.

    - Move First
    - Move First, Update
- **After Move Next** – Triggered whenever a Move Next command is executed.
    - Move Next
    - Move Next, Update
- **After Move Previous** – Triggered whenever a Move Previous command is executed.
    - Move Previous
    - Move Previous, Update
- **After Move Last** – Triggered whenever a Move last command is executed.
    - Move Last
    - Move Last, Update
- **After Move End** – Triggered whenever a Move End command is executed.
- **After Start of Data** – Triggered whenever the Data Object navigates past the first record.
- **After End of Data** – Triggered whenever the Data Object navigates past the last record.
- **Timer Started** – Triggered whenever the Data Object timer is started.
- **Timer Stopped** – Triggered whenever the Data Object timer is stopped.