PRIME Parameters, Expressions, and Conditions User Guide Version 5.1

March 2025



Chyron PRIME Parameters, Expressions, and Conditions User Guide • 5.1 • March 2025 This document is distributed by Chyron in online (electronic) form only, and is not available for purchase in printed form.

This document is protected under copyright law. An authorized licensee of Chyron Audio Director may reproduce this publication for the licensee's own use in learning how to use the software. This document may not be reproduced or distributed, in whole or in part, for commercial purposes, such as selling copies of this document or providing support or educational services to others.

Product specifications are subject to change without notice and this document does not represent a commitment or guarantee on the part of Chyron and associated parties. This product is subject to the terms and conditions of Chyron's software license agreement. The product may only be used in accordance with the license agreement.

Any third party software mentioned, described or referenced in this guide is the property of its respective owner. Instructions and descriptions of third party software is for informational purposes only, as related to Chyron products and does not imply ownership, authority or guarantee of any kind by Chyron and associated parties.

This document is supplied as a guide for Chyron Audio Director. Reasonable care has been taken in preparing the information it contains. However, this document may contain omissions, technical inaccuracies, or typographical errors. Chyron and associated companies do not accept responsibility of any kind for customers' losses due to the use of this document. Product specifications are subject to change without notice.

Copyright © 2025 Chyron, ChyronHego Corp. and its licensors. All rights reserved.



Table of Contents

Parameters/Expressions	5
Introduction	5
Parameter Scopes	6
Application Parameters	6
Project Parameters	6
Scene Parameters	6
PRIME Playout	7
Add New Parameter	7
Remove Parameter	7
Save Parameters	8
Click the Save button to save all the parameters in the selected scope (either project or application) as an XML file	8
Import Parameter XML file	8
Click the Import button to Load or Append a Parameter XML file	8
Edit Parameter	8
PRIME Editor	9
Adding New Parameter	9
Remove Parameter	9
Link Parameter	9
Import Parameter XML file	9
Click the Import button to Load or Append a Parameter XML file	9
Save Parameters	9
Edit Parameters	. 10
Disable / Enable Parameter	.10
Modify Parameter Scope	. 10
Modify Parameter Order	. 10
Parameter Type	11
Parameter Type Summary	. 11
Parameter List	12
Parameter Name	12
Parameter Value	. 12
Sample Value Conversion	.13
Bindings	. 13
Manual Bindings	. 14
Drag-and-Drop Binding	. 15



Option 1, Dragging from the Properties Panel	15
Option 2, Dragging from the Keyframe Panel	17
Option 3, Dragging from the Scene Tree	17
Expressions	17
Expression Text	24
Expression Language Support	24
Logical Operators	24
Comparison Operators	24
Arithmetic Operators	25
String Operators	26
Mathematical Functions	26
Text Functions	27
Specialized Keywords	28
Other Functions	31
Expression Evaluation	32
Advanced Bindings	32
Conditions	33
Introduction	33
Creating a Condition	33
Adding a Comment within a Condition, Logic, or Stylesheets	40
Triggering a Condition	41
Show what triggers a condition	42



Parameters/Expressions

Introduction

PRIME supports both parameters and expressions.

At its most basic level, a **parameter** is a container for a specific type of data that may be leveraged by other elements of a scene. Parameters reference values directly; e.g. an Integer parameter that is set to the number 10. The value of this parameter will never change from 10 unless the parameter itself is changed.

On the contrary, an **expression** may be thought of as a formula that is evaluated dynamically; e.g. an Integer expression may be set to the X position of an object in the scene. As a result, the value of the expression will automatically change as the object is moved along the X-axis. Scenes have their own distinct collection of parameters and expressions that are transient in nature. This means that initial values that are set while working with the designer are lost once the scene has been cleared. Each project maintains a collection of parameters, and updates to these parameters, that persist in real time. These parameters are global to all scenes within the project.



Parameter Scopes

Application Parameters

The scope of Application Parameters is global meaning this parameter type will be accessible to every Project and every scene within every project.

Project Parameters



Project parameters are only accessible to every scene in the current project the parameter is created in.

Scene Parameters

Scene parameters are only accessible to the individual scene they are created in.





PRIME Playout

In Prime Playout you are able to view Application and Project Parameters.

To view Application parameters as well as the Project parameters of the currently active project select **View > Parameters**.



Add New Parameter

Clicking the **Add** button will create a new parameter at the scope you have selected, either Application or Project. The default type of the new parameter is String (text) with a null default value.



Click the down arrow, to the right of the **Add** \downarrow button to select a different parameter type to add to the scope selected, either Application or Project.

See Parameter Type Summary for more information

Remove Parameter

Clicking the **Remove** button will delete the currently selected parameter from the scope you have selected (either project or application).



Save Parameters

Click the **Save** button to save all the parameters in the selected scope (either project or application) as an XML file.

Import Parameter XML file



Click the **Import** button to Load or Append a Parameter XML file.

Select Load to only display parameters saved in the XML file.

Select Append to add parameters in the XML file to the current display of

parameters.

Edit Parameter

Click Parameter type icon to change the type of a parameter



To edit the name or value of a Parameter click into the cell you wish to update.





PRIME Editor

Adding New Parameter

Clicking the **Add** button will create a new scene parameter. The default type the new parameter is String (text) with a null default value.



Click the down arrow, to the right of the **Add** \downarrow button to select a different parameter type to add to the scope selected, either Application or Project.

See Parameter Type Summary for more information

Remove Parameter

Clicking the **Remove** button will delete the currently selected parameter.

Link Parameter

Clicking the **Link Parameter** button lets you navigate to an existing Project or Application parameter that can then be linked to the active scene.

Import Parameter XML file



Click the Import button to Load or Append a Parameter XML file.

Select Load to only display parameters saved in the XML file.

Select **Append** to add parameters in the XML file to the current display of

parameters.

Save Parameters

Click the **Save** button to save parameters as an XML file.



Edit Parameters

😢 💶	Name	Value	Bindings
🖸 🃮	abe Parameter 1	Hello 124	Text1.Text
-	abe Project Paramter		Text2.Text
	abe Applicaton Parameter		

Disable / Enable Parameter



By default any new parameter will be enabled. Press the negative space in the enable column to disable the selected parameter.

Modify Parameter Scope



Click the scope icon of the parameter you wish to modify. This will reveal a list of available scopes to select from; Scene, Project and Application.

Modify Parameter Order



Hold down mouse click left, on the negative space to the left of the Parameter name. Drag and drop parameter to desired position in the list.



Parameter Type



Click the parameter type icon of the parameter you wish to modify. This will reveal a list of available parameter types. Modifying the type of an existing parameter with a set value will attempt to convert the current value to one appropriate for the new type. If the data cannot be converted, then the new value will be defaulted accordingly.

Parameter Type Summary

Туре	
String	Sequence of Alphanumeric characters
Boolean	Can only have one of two values, True or False
Integer	Whole Numbers without decimals
Color	Color [Black]
Float	Stores Fractional Numbers with one or more decimal
ByteArray	Integers in the range between 0 and 255
DateTime	1/1/0001 12:00:00 AM
Double	Fractional numbers. Sufficient for storing 15 decimal digits
Long	Whole numbers from -9223372036854775808 to 9223372036854775807. Used when int is not large enough
TimeSpan	Time interval and can be expressed as a particular number of days, hours, minutes, seconds, and milliseconds





Parameter List

A Parameter list is a container, where parameters within the same scope can be "grouped". This allows you to organize parameters together, rather than having a single long list of parameters.

To add a new parameter into a Parameter list press the + icon to the right of the parameter list. Existing parameters can be dragged and dropped into a Parameter List.



You can only add parameters into a parameter list that have the same scope.

Parameter Lists can be nested by drag and dropping a Parameter list into another Parameter list.

Parameter Name

Click in the parameter name field to edit the parameter name. Each parameter must have a unique name. However, uniqueness is only guaranteed within the context of the project parameter collection. Scenes may include parameters with names that match project parameters because these are addressed differently.

😧 💶	Name	Value	Bindings
	abc Parameter 1	Hello	Text1.Text

Parameter Value

Туре	Default
String	Empty string
Boolean	False
Integer	0
Color	Color [Black]



Float	0
ByteArray	Empty byte array
DateTime	1/1/0001 12:00:00 AM
Double	0.0
Long	0
TimeSpan	TimeSpan with 0 Ticks

Sample Value Conversion

Whenever possible, existing parameter values will be converted.

Initial Type	Current Value	Modified Type	New Value
Integer	1	Double	1.0
Integer	1	Boolean	True
Integer	0	Boolean	False
String	1	Double	1.0
String	Hello	Double	0
Double	1.7	Integer	1

Bindings

Parameter values may be bound directly to one or more object properties within a scene. These bindings appear in the **Bindings** column and may be modified either by direct text entry into the bindings field, or through a drag-and-drop operation. Scene parameters may only be bound to objects within the same scene. However, project parameters may be bound to objects of any scene within the project. Project parameters may be viewed and modified in the designer by clicking the **Project** text within the parameters panel.

Bindings allow a parameter to automatically update one or more targets whenever the parameter value is changed. For example, the user may set up an Integer parameter that is bound to the opacity of an object within the scene. When the parameter is updated, the opacity of the bound object will update as well.



Manual Bindings

The user may bind a parameter directly to an object by clicking the binding field within the parameter grid control and typing text that describes a valid target. Target text has several variations, but all conform to two standards: (1) A single period character separates fields and (2) names are trimmed to remove white space. For example:

Target Type	Scene Example
Object Property	Text1.Text
Keyframe Property	Text1.Action1.Keyframe1.PositionX

Notice that while an action may be named **Action 1**, the target text removes the space and uses **Action1 instead**. In general, a scene parameter can either be bound to a property with the form *ObjectName.PropertyName* or to a specific property of a keyframe using the form *ObjectName.ActionName.KeyframeName.PropertyName*. Because scene parameters are actual objects within the scene, they are also valid binding targets. This means that one parameter can be bound to the value of another (example target text: Parameter1.Value). Project parameters add one stipulation to bindings in that their bindings may only target scenes explicitly or other project parameters.

Target Type	Project Example
Object Property	SceneName.Text1.Text
Keyframe Property	SceneName.Text1.Action1.Keyframe1.PositionX
Project Parameter	Project.Parameter1



Drag-and-Drop Binding

To facilitate binding object properties to parameters, the user may utilize drag-and-drop from three locations within the designer.

Option 1, Dragging from the Properties Panel

In the property panel, hover over the property you wish to bind the parameter to. Holding left click on the mouse drag and drop the individual item directly onto the Parameters panel.





Alternatively, the property can drop onto the parameter panel itself. This will create a new parameter already bound to the property being dragged and with a parameter type that matches the type of the property.



16 | PRIME 5.1 Parameters, Expressions, and Conditions User Guide



Option 2, Dragging from the Keyframe Panel

Similarly, the user may drag properties directly from the Keyframe panel.



Option 3, Dragging from the Scene Tree

The user may also drag objects from the scene tree directly onto the Parameters panel. This will generate target text for bindings based on the default property of the object dragged. For example, dragging a text object will utilize the Text property (example target text: Text1.Text) while dragging a file picker control will utilize the File property (example target text: FilePicker1.File).





Expressions

Expressions are specialized parameters that evaluate dynamically, rather than having an explicit set value. Instead, the value is calculated based on a formula the user defines. Once configured, expressions support bindings in the same manner as parameters.

To view the expressions within a scene, click the **Expressions** text in the Parameters panel or click the Expressions node in the scene tree.

🜉 ChannelBox Prin	ne Scene Editor - New Sce	ene 1.	pbx							-		×
File Edit View	Window Tools	Help								CHYR	ONHE	GO
Project TV		\sim	Canvas 📄 Con	ntrol Panel 📡 Script	ting						🕎 Pla	yout
Toolbox Scenes T	ransitions		↔ Move 🖸 Scale 🕜	🔖 Rotate 🍰 Anchor	💥 Delete		🕭 World 🖪 Auto Sele	ct Propert	ties Events			
A Text		î						⊠a	A Text 1		a	î
S Clip	III Video Input							> Ren	der			
i Group	Cube							✓ Tran	sform			
🚳 Model								Posi	tion X 0.0 🜩 Y 507	.5 ‡ Z 0.0	÷	
Effects	•	^	lext 1					Scale	e x 1.00 🛋 v 1.0	0 1 7 1.00		
Auto Follow	Crop							D				
Mask	🗱 Material							Kota	tion X 0.0 - Y 0.0	₹ 2 0.0	-	
V Light	Render Texture							Pivo	t X 0.0 - Y 0.0	÷ Z 0.0	÷	
Warp	Page Turn							✓ Surf	face			
ransition	Clawi	~						Size	Width 960 🖨	Height 135	÷ •	
Scene Tree		×	Parameters					× Text	Width 0	Height 0	-	
» 🖻 🗎 🦻	*		Descient	Add St Remove				Dopad	city 100.0			
Objects	Effects		Project	Name	Type	Value	Bindings					
V Scene Gro	up	Г	A# Parameters	Numeric Parameter	Integer	• 0	Text1.PositionX	✓ lext	2000		-	
aA Text 1			f(x) Expressions	Parameter 1	Double	▼ 507.5	Text1.PositionY	Style	Arial 100 (Text 1)	~		
 Resources A# Parameter 	5	L	45	-1				Font	Arial		~	
Numer	ic Parameter							Size	100.0			
f(x) Expression	eter 1 Is											~
E Control Pa	anel		Timeline					× Keyfran	ne			×
E Scripting			Default 👒 Add Act	tion			Default	~ Nam	ne (Cursor)	Frame	00:00:00.00	A
			Action 🔶 🐋 🛸	Triggered By (0)	▶ ■ H ↔ I	Zoo	m 🔎 🚃 🛃 🔎	>> Teine				
			Animation 7	0 1:00 2:0		4:00 5:00	6:00 7:00	ing <u>e</u>				
			Text 1					✓ Prop	perties			
								Nar	me Value In	Out		
												_





Expressions may be added, modified or removed in the same manner as scene or project parameters. However, the Value field of each expression is read only. Instead, a new field entitled Expression is available. The text in this field defines the formula that will be evaluated to determine the current value of the expression.



As with bindings, the expression text utilizes a naming convention for incorporating object and keyframe properties. As in the above screenshot, the user may set the expression text to an object property (Text1.Width) and once the edit has been committed, the Value column will update immediately to reflect the new value (960).



Parameters							×
🚺 Project	中 Add 💥 Rem	ove					2
A# Parameters	Name	Туре		Expression		Value	Bindings
7 m r diameters	Expression 1	Double	-	Text1.Width * 2	T	960	
f(x) Expressions							

Unlike simple bindings, expression text can include a variety of mathematical, logical and string operations. Each expression can include references to one or more constant values (1, True, "ABC") or variables (Text1.Width). For example, in the screenshot above the expression is being modified to multiply the width of *Text 1* by the number 2.

Parameters						×
🚺 Project	t 🕂 Add 💥 Remove 🕜					
A# Parameters	Name	Туре	Expression		Value	Bindings
Pam Functions	Expression 1	Double 🔹	Text1.Width * 2	N	1920	
f(x) Expressions				NS		
				- v		

This evaluates to 1920 as indicated by the value column. As with bindings, the user may drag-and-drop directly onto the expression column of an existing expression.



ChannelBox Prime Scene Editor - New Scene 1.	pbx	– 🗆 X
File Edit View Window Tools Help		CHYRONHEGO
Project TV 🗸	📉 Canvas 🧮 Control Panel 🐺 Scripting	Playout
● Project IV Toolbox Scenes Transitions Graphics ▲ Text ■ Image ③ Clip ■ Video Input ③ Group ③ Model Effects ■ Mask ∞ Material	Move Scale Rotate Anchor Control rane Auto Select	Properties Events * Render * Transform * Position X 0.0 ÷ Y 507.5 ÷ Z 0.0 ÷ Scale X 100 ÷ Y 100 ÷ Z 0.0 ÷ * Rotation X 0.0 ÷ Y 0.0 ÷ Z 0.0 ÷ *
💡 Light 🛛 🖾 Render Texture		Pivot X 0.0 🕈 Y 0.0 🗣 Z 0.0 🗣
Warp 📝 Page Turn		✓ Surface
Transition 🖉 Crawl 🗸		Size Width 960 🜩 Height 135 🜩 🛎
Scene Tree X	Paramaterr	Text Width 0 🜩 Height 0 🐳
<u>* 0 0 v 7 × 1</u>		Opacity 100.0
Objects Effects New Scene 1 Scene Group A Text 1 P Resources Numeric Parameter Parameter 1 Numeric nameter 1	Project Att Parameters Name Type Expression Value Bindings f(x) Expressions String Image: Comparison of the string of th	✓ Text Style Arial 100 (Text 1) Text Size 100.0 Text Text
Expression 1	Timeline ×	Keyframe X
Control Panel	Default > Add Action Default > Action → Triggered By (0) → M	Name (Cursor) Frame 00:00:00:00 Image: Constraint of the second se

If the expression text is currently empty, the property will be added by itself. Otherwise, the property will be appended with an addition operator.



							_
Channelbox Prime Scene Editor - New Scene I	.pbx						ò
File Edit View Window Tools Help						CHYRONIEG	JU
Project TV ~	Canvas 🔡 Co	ntrol Panel Scripting		_		Play	out
Toolbox Scenes Transitions	↔ Move 🖸 Scale (🗞 Rotate 🌛 Anchor 🛛 💥 Dele	te	🕭 Worl	d 🖳 Auto Select	Properties Events	
A Text						🗹 🗚 Text 1 🗃	î
Clip Video Input						> Render	
Soup Cube						✓ Transform	
🔏 Model						Position X 0.0 A Y 507.5 A 7 0.0 A	
Effects	Text 1		٦				
🚥 Auto Follow 🔲 Crop	•						
🖾 Mask 🛛 🗱 Material						Rotation X 0.0 + Y 0.0 + Z 0.0 +	
💡 Light 🛛 🔤 Render Texture						Pivot X 0.0 🔹 Y 0.0 🔹 Z 0.0 🔹	
🔛 Warp 📝 Page Turn						✓ Surface	
💽 Transition 🛛 🖨 Crawl 🗸	·					Size Width 960 🌩 Height 135 🌩 🖬	
Scene Tree X						Text Width 0 🚔 Height 0 🚔	
🍾 🖻 💼 🤛 🦉 🗶	Parameters				×		
Objects Effects	Project	📫 Add 💢 Remove			8		
New Scene 1	A# Parameters	Name Type	Expression	Value	Bindings	✓ Text	
Text 1	f(x) Expressions	Expression 1 Double	• Text1.width	900		Style 🗛 Arial 100 (Text 1) 🗸 🔚 🕅	
Presources		1			3	Font Arial	
All Parameters Numeric Parameter						Size 100.0	
Parameter 1							~
f(x) Expressions	Timeline				x	Keyframe	×
E Control Panel	Default 👒 Add Ad	tion		Def	fault 🗸		
E Scripting	Action In In	Triggered By (0)	M 44 NA M	700m @		Name (Cursor) Frame 00:00:00.00	
						Triggers	r i
	Animation 0:	00 1:00 2:00 3	:00 4:00	5:00 6:00	7:00	✓ Properties	
	Text 1					Name Value In Out	1
						1	

The user may also drag directly onto the grid control to create a new expression altogether.

🜉 ChannelBox Prime Scene Editor - N	Vew Scene 1.pbx							– 🗆 X
File Edit View Window To	ools Help						(HYRONHEGO
Project TV	 Canva 	as 📕 Control Panel	Scripting					🗾 Playout
Toolbox Scenes Transitions	🕂 Move 🔃 Scale 📢	Rotate 🔑 Anchor	💥 Delete		🕭 World	🗬 Auto Select	Properties Events	
A Text					_		🗹 🛋 Text 1	🧠 <u><u></u></u>
🖓 Clip 🛤 Video In							> Render	
Group 🗍 Cube							✓ Transform	
🚳 Model							Position X 0.0 A Y 507.5 A	7 0.0
Effects	Text 1							7 100
🚥 Auto Fo 🔲 Crop								2 1.00 -
🔲 Mask 🛛 🗱 Material							Rotation X 0.0 🜩 Y 0.0 🜩	Z 0.0
💡 Light 🛛 🖾 Render							Pivot X 0.0 🛉 Y 0.0 🔹	z 0.0 ≑
🔛 Warp 📝 Page Tu							✓ Surface	
Transition 🜾 Crawl 🗸					_		Size Width 960 🖨 Heigh	t 135 🜩 🔺
Scene Tree X	Parameters				_	×	Text Width 0 🜩 Heigh	t 0 🜩
s 🗈 🗈 🔊 🖉 🗶							Opacity 100.0	
Objects Effects	Project	Name	Tune	Expression	Value	Bindings		T
New Scene 1 Scene Group	A# Parameters	Expression 1	ouble 🔹	Text1.Width * 2	1920	bindings	V Text	_
aA Text 1	f(x) Expressions						Style Arial 100 (Text 1)	~ 🗖 🖉
Resources All Parameters							Font Arial	~
Numeric Parameter							Size 100.0 🜩	
Parameter 1								v
Expression 1	Timeline					×	Keyframe	×
Control Panel	Default 👒 Add Acti	on			Defaul	t v	Name (Curror)	Carran 00-00-00 00 🛋
E Schpung	Action 🔶 🗮 👒 T	riggered By (0) 🏮	• = H	Keyframe Zoo	m 🔎 🚃	📕 🔎 🔭	(cusor)	
	Animation				100		Triggers	Ŧ
	Text 1	1.00 2.00	3.00	4.00 3.00 0.00	7.00	0.00 5.1	✓ Properties	
							Name Value In	Out



ChannelRey Drime Scene Editor	Jaw Scene 1 nhv						
Eile Edit View Window To	vew scene n.pox						0
Project TV	Canvi	as 🗮 Control Papel 😤 Scripting					
Project TV Toolbox Scenes Transitions Graphics ▲ Test Image © Clip Wideo In ▲ Clip © Group © Cube Model Effects Effects ▲ ▲ ▲ Mask W Material ▲	✓ Move Scale	ss ∭ Control Panel ≫ Scripting Rotate ≱ Anchor X Dekte		Norld 🗟	Auto Select	Properties Events)ut
V Light Render						Pivot X 0.0 + Y 0.0 + Z 0.0 +	
Warp Page Tu Transition Crawl						✓ Surface	
Scene Tree X						Size Width 960 🔹 Height 135 🔹 🗎	
5 A A A A A A A A A A A A A A A A A A A	Parameters				×	Text Width 0 😴 Height 0 😴	
Objects Effects	Project	📲 Add 💥 Remove			0	Opacity 100.0 🖨	
New Scene 1	A# Parameters	Name Type	Expression	Value Bi	lindings	V Text	
✓ Scene Group aA Text 1	f(x) Expressions	Expression 1 Double	Text1.Width * 2	1920		Style 🖓 Arial 100 (Text 1) 🗸 🚽 💟	
Resources Parameters Numeric Parameter Services Numeric Parameter Services Numeric Parameter Services Numeric Parameter Services Parameter Services Services						Font Arial ~	~
Expression 1	Timeline				×	Keyframe	×
Control Panel	Default 👒 Add Acti	ion		Default	~	Name (Cursor) Frame 00:00:00.00 +	
	Action 🔶 🚖 🛸 T	Triggered By (0) 🕨 🔳 🔣 📢 🕅	Keyframe Zoom	n /P	··· / / /	Triggers	í.
	Animation 0:00	0 1:00 2:00 3:00 4	11	7:00 8:0	.00 9:(✓ Properties	_
	Text 1					Name Value In Out	

The expression text will be set to the property being dragged and the type will match accordingly.

ChannelBox Prime Scene Editor - N	New Scene 1.pbx									-		٦
File Edit View Window To	ools Help								C	IYRC	NHEGO)
🚺 Project TV	V 📉 Canva	es 📄 Control Pan	el 🔂 Scripting								Playout	ł
Toolbox Scenes Transitions Graphics	🕂 Move 🖾 Scale 📢	Rotate 🔑 Anchor	💥 Delete		🕭 World 🛽	🖏 Auto Select	Properties	Events	-		~	4
aA Text 🔳 Image							⊠ a ¹ [Text 1				
Sclip 🔤 Video In							> Render					
i Group 🗍 Cube							✓ Transfo	orm			_	
Model	Text 1						Positio	n X 0.0 🌩 Y	507.5 🛊	Z 0.0	÷	
Effects							Scale	X 1.00 후 Y	1.00 🜲	Z 1.00	÷ 🔒	
Mask Material							Rotatio	n X 0.0 후 Y	0.0	z 0.0	•	
Vight 🖾 Render							Pivot	X 0.0 🜩 Y	0.0	z 0.0	÷	
🔛 Warp 📝 Page Tu							v. Surface					
🖭 Transition 🛛 🖨 Crawl 🗸							• Surraci	Width 060	Height	135		
Scene Tree X							5126	Width 0	i leight	0	-	
% 🗈 🗈 🗩 🖉 🗶	Parameters					×	Text		Height			
Objects Effects	Project	🕂 Add 💥 Remov	/e			0	Opacity	100.0 🖨				
New Scene 1	A# Parameters	Name	Туре	Expression	Value	Bindings	✓ Text -					
✓ [™] Scene Group	f(x) Expressions	Expression 1	Double	Text1.Width * 2 Text1 Text - N	1920		Style	Arial 100 (Text 1)		~	2	
P Resources		expression 2	Sang	· Textilient			Font	Arial		_	~	
All Parameters Numeric Parameter							Size	100.0			-	
Parameter 1							JALC		.			,
Expressions	Timeline					×	Keyframe				×	
Expression 2	Default 👒 Add Acti	on			Default	~						
Control Panel	Action 🔶 🚖 🖷 T	riggered By (0)	► ■ N 44	Keyframe	Zoom 🔎	🕨 🔎 📕	Name	(Cursor)	Fra	ime 00	:00:00.00	
	Animation	ntrolontro	huutuuluut	mhantaohaata	alantaalanta	uluutuul	Trigger	5		1	a tee Window both Hog angesty	
	Text 1	1:00 2:	00 3:00	4:00 5:00	6:00 7:00	8:00 9:0	✓ Proper	ties		A hat B Cy	They See .	Í
	Text						Name	Value	In	Out	- Dire	
										Cafet Cafet Mary	H Manid Hi Tanto	l
										have been	P # X	
										0 Open	fram 1 Ad Personales	1



Expression Text

Expression text is evaluated left to right following an order of operations that is dependent upon the type of operation occurring within the expression. This simply means that evaluation is left to right except in the case of parentheses and mathematical operators.

Expression Text	Evaluation
7	Evaluates to the number 7.
Text1.Width	Evaluates to the current width of Text 1.
7 * Text1.Width	Evaluates to the number 7 multiplied by the current width of Text 1
7 * (Text1.Width + 3)	Evaluates to the number 7 multiplied by the result of adding 3 to the current width of Text 1.
Text1.PositionX + Text1.Width	Evaluates to the sum of Text 1's X-position and width.

Expression Language Support

Expressions support a number of logical, comparison, arithmetic and text operators as well as many mathematical functions.

Lo	aica	perators
LU	giuu	perators

Operator	Usage	Meaning
and, &&	X and Y X && Y	True if both the left and right operand are true. False otherwise. <i>This may only be used to compare Boolean values</i> .
or,	X or Y X Y	True if either the left or right operand is true. False otherwise. <i>This may only be used to compare Boolean values.</i>
xor, ^	Х хог Х Х ^ Ү	True if only one of the operands is true. False otherwise. <i>This may only be used to compare Boolean values.</i>
not, !	not(X) !X	True if the operand is false. False otherwise. This may only be used to compare Boolean values.



Comparison Operators

The following operators may be used to compare two values within the expression.

Operator	Usage	Meaning
<	X < Y	True if the left operand is less than the right operand. <i>This may only be used to compare numeric values.</i>
<=	X <= Y	True if the left operand is less than or equal to the right operand. <i>This may only be used to compare numeric values.</i>
=	X = Y	True if both operands are equal.
!=	X != Y	True if both operands are not equal.
>	X > Y	True if the left operand is greater than the right operand. <i>This may only be used to compare numeric values</i> .
>=	X >= Y	True if the left operand is greater than or equal to the right operand. <i>This may only be used to compare numeric values.</i>



Arithmetic Operators

The following operators may be used to perform basic arithmetic on numeric data values.

Operator	Usage	Meaning
+	X + Y	Performs an addition operation. This may only be used with numeric values.
-	X - Y	Performs a subtraction operation. <i>This may</i> only be used with numeric values.
*	X * Y	Performs a multiplication operation. <i>This</i> may only be used with numeric values.
/	X / Y	Performs a division operation. <i>This may</i> only be used with numeric values.
Mod	X Mod Y	Performs a modulus operation. This may only be used with numeric values.
٨	Χ ^ Υ	Raises the left operand to the exponential power denoted by the right operand. <i>This may only be used with numeric values.</i>
- (Unary)	-X	Multiplies the given number by -1.

String Operators

Operator	Usage	Meaning
+	X + Y	Concatenates two text strings together. Literal text values must be enclosed within double quotation marks (e.g. "ABC" + "DEF").



Mathematical Functions

Function	Usage	Meaning
Maximum	Math.Max(X, Y)	Returns X or Y depending upon which value is larger.
Minimum	Math.Min(X, Y)	Returns X or Y depending upon which value is smaller.
Sign	Math.Sign(X)	Returns a value indicating the sign of X.
Absolute	Math.Sign(X)	Returns the absolute value of X.
Arc Sine	Math.Asin(X)	Returns the angle whose sine is X.
Arc Cosine	Math.Acos(X)	Returns the angle whose cosine is X.
Arc Tangent	Math.Atan(X)	Returns the angle whose tangent is X.
Hyperbolic Cosine	Math.Cosh(X)	Returns the hyperbolic cosine of angle X.
Hyperbolic Sine	Math.Sinh(X)	Returns the hyperbolic sine of angle X.
Hyperbolic Tangent	Math.Tanh(X)	Returns the hyperbolic tangent of angle X.
Square Root	Math.Sqrt(X)	Returns the square root of X.
Cosine	Math.Cos(X)	Returns the cosine of angle X.
Sine	Math.Sin(X)	Returns the sine of angle X.
Tangent	Math.Tan(X)	Returns the tangent of angle X.



Text Functions

Function	Usage	Meaning
Contains	Text1.Text.Contains("ABC")	Returns true if the text value "ABC" exists anywhere in the text value of Text1.Text.
EndsWith	Text1.Text.EndsWith("ABC")	Returns true if the text value of Text1.Text ends with the text "ABC"
IsEmpty	Text1.Text.IsEmpty()	Returns true if the text value of Text1.Text is either null, empty or all whitespace.
Length	Length(Text1.Text)	Returns the number of characters in the text value of Text1.Text
ReadFile	ReadFile("C:\Sample.txt")	Opens a text file and returns all lines.
Replace	Text1.Text.Replace("a", "b")	Replaces all instances of the text value "a" with the text value "b" in Text1.Text and returns the resulting value
StartsWith	Text1.Text.StartsWith("ABC")	Returns true if the text value of Text1.Text starts with the text "ABC"
Substring	Text1.Text.Substring(0, 5)	Retrieves a substring from Text1.Text
ToLower	Text1.Text.ToLower()	Returns a copy of Text1.Text converted to lowercase.
ToUpper	Text1.Text.ToUpper()	Returns a copy of Text1.Text converted to uppercase.
Trim	Text.Trim()	Returns a copy of Text1.Text with all leading and trailing whitespace removed.



Specialized Keywords

While expressions may interact with objects within a scene or execute global functions, they may also reference objects accessible through keywords that are context specific.

Channel: Refers to the channel upon which the current scene is loaded. Evaluating items that reference the Channel keyword is only supported in a playout environment; e.g. scenes open in the designer do not have a parent channel.

ltem	Usage	Meaning
Index	Channel.Index	Returns the numeric index of the channel in which the executing scene is loaded.
Name	Channel.Name	Returns the text name of the channel in which the executing scene is loaded.
CloseScene	Channel.CloseScene(SceneName) SceneName: Text	Attempts to close a scene with the specified name or file path. This will only affect the channel in which the executing scene is loaded.
Is Description And Layer On Output	Channel.IsDescriptionAndLayerOnOutput(Desc ription, Layer) Description : Text Layer : Specific number or text like "1-3"	Returns a value indicating whether a scene with the specified description is loaded currently on the specified layer within the current channel of the executing scene. The matching scene must be different from the executing scene.
Is Description On Output	Channel.IsDescriptionOnOutput(Description) Description: Text	Returns a value indicating whether a scene with the specified description is loaded currently within the current channel of the executing scene. The matching scene must be different from the executing scene.
lsLayerOnOutput	Channel.IsLayerOnOutput(Layer) Layer: Specific number or text like "1-3"	Returns a value indicating whether a scene is loaded currently on the specified layer within the current channel of the executing scene. The matching scene must be different from the executing scene.
IsSceneAndLayerOnOutput	Channel.IsSceneAndLayerOnOutput(Name, Layer) Name : Text Layer : Specific number or text like "1-3"	Returns a value indicating whether a scene with the specified name is loaded currently on the specified layer within the current channel of the executing scene. The matching scene must be different from the executing scene.



IsSceneOnOutput	Channel.IsSceneOutput(Name) Name : Text	Returns a value indicating whether a scene with the specified name is loaded currently within the current channel of the executing scene. The matching scene must be different from the executing scene.
LoadScene	Channel.LoadScene(SceneName) SceneName: Text	Attempts to load a scene with the specified name or file path. This will only affect the channel in which the executing scene is loaded.
PlayScene	Channel.PlayScene(SceneName) SceneName: Text	Attempts to play a scene with the specified name or file path. This will only affect the channel in which the executing scene is loaded.
StopScene	Channel.StopScene(SceneName) SceneName: Text	Attempts to stop a scene with the specified name or file path. This will only affect the channel in which the executing scene is loaded.

Project: Refers to the parent project of the current scene. Currently this keyword is only used to access parameters defined within the project.

For example, *Project.Parameter1* returns the value of a parameter named *Parameter 1* that exists within the parent project.

Item	Usage	Meaning
Layer	Scene.Layer	Returns the layer currently assigned to the scene.
Loaded	Scene.Loaded	Returns a value indicating whether the scene is currently loaded onto a channel.
Playing	Scene.Playing	Returns a value indicating whether the scene is currently playing on a channel.
Name	Scene.Name	Returns the name of the current scene.
Close	Scene.Close	Closes the current scene if it's loaded or playing on a channel.
PlayAction	Scene.PlayAction(Name)	Plays an action with the specified name.
Play	Scene.Play	Plays the current scene if it's loaded; if the scene is already playing then this will have no effect.

Scene: Refers to the current scene.



Stop	Scene.Stop	Stops the current scene if it's playing; if the
		scene isn't playing then this will have no
		effect.

Other Functions

Operator	Usage	Meaning
GetObjectValue	Scene.GetObjectValue(Name, Property)	Returns the value of a specific property defined within a target object.
	Name: Scene object name Property: Property name	GetObjectValue("Image1", "File")
SetObjectValue	Scene.SetObjectValue(Name, Property, Value)	Applies a value to a specific property defined within a target object.
	Name: Scene object name Property: Property name Value: Value to assign	SetObjectValue("Image1", "File", "C:\\Sample.png")



Expression Evaluation

Only after the entire expression has been evaluated is the result converted to a value appropriate for the given expression type. Consequently, expressions may include mixed data types as long as the result fits the constraints of the expression type. For example:

Expression Type	Expression Text	Evaluation	Value
Double	1 + 7	8	8.0
Integer	1 < (30 – 7)	1 < (23) = True	1
Integer	"10" + "30.2"	"1030.2"	1030
Boolean	(30 < 7) and True	(False) and True	False

Parentheses may also be used to explicitly denote order of operation.

Advanced Bindings

Expressions can also be used when setting up bindings. Instead of simply specifying a target (e.g. **Text1.Text)**, the user may use the syntax below to incorporate expressions.

Text1.Text = Expression

Expression is any valid expression text with the additional variable available (VALUE) which evaluates the value of the parameter that has changed. This expression text would be manually entered into the binding column for a parameter or expression. For example, suppose an Integer parameter was configured with the binding expression below: Text1.Width = value * 10

If the parameter was updated with value 5, then the binding would immediately result in a value of 5 * 10 getting applied to Text1.Width.



Conditions

Introduction

Condition objects may be utilized to build logic into the execution of event triggers. Conditions appear in the Scene Tree in the same manner as *parameters* and *expressions*, however, a different panel is used to construct and modify them.



Clicking on the *Conditions* node in the Scene Tree or the Conditions item in the View Menu will display the Logic pane seen below. The Logic pane will display Conditions, Logic effects and Style Sheet effects.





Creating a Condition

The first step to incorporating conditional logic into a scene is to add a new Condition object. This can be achieved by clicking the **Add (+)** button in the editor panel.



This will create a new, empty condition. The Condition may be renamed by clicking the top-level item in the tree. **Names must be unique**.

Parameters / Expression	ons Logic			
 Conditions Logic Style Sheet 	+ ¥ ⊶ - ⓑ ¥ Condition 1	Statements IF Bill elle 🏹	Commands 🕤 Trigger 🔳 Property 🛛 💥	** 📕 Evaluate 🛛 Clear Status

Once the desired name has been decided upon, the next step is filling out the conditional tree structure. The currently supported conditional statements are:

- If This tests a Boolean (*True/False*) expression. If the expression evaluates to *True*, then the statements contained within the If will be evaluated as well.
- **Else If** This tests a Boolean (*True/False*) expression as well, however this statement type may only exist following an **If** statement. If the expression evaluates to *True*, then the statements contained within the **Else If** will be evaluated as well.
- Else This statement type may only exist following an If or Else If statement. The statements contained within the Else will evaluate if the associated If and Else If statements all evaluated to *False*.



• While Loop - In while loop, a condition is evaluated before processing statements inside the loop. If a condition is true then and only then the body of a loop is executed.

After the body of a loop is executed then control again goes back to the beginning, and the condition is checked if it is true, the same process is executed until the condition becomes false.

Once the condition becomes false, the control goes out of the loop.



Outside while loop

In a while loop, if the condition is not true, then statements inside the loop will not be executed.

Example of using a while loop condition would be to loop through a variable quantity of items in a data table column, and have that list of items populate a control panel drop down list.

Parameters / Expressions Logic		
22 Conditions	Statements If etf Esc Commands Trigger Property X Image: Statements Data1.MoveFirstUpdate() Image: Statements Image	* 🖡 Evaluate 🛛 Clear Status

Advanced:

- Timeout: Prevents the while loop from remaining in a continuous loop, and will timeout after the specified duration. This default is 1 second.
- Asynchronous: Allows all statements within the condition to evaluated at the same, versus being evaluated one at a time
 *Conditions by default run synchronously
- **Trigger** This statement type may exist independently of an **If/Else If/Else** test block, or nested within any **If/Else If/Else** item. This defines a list of triggers that will execute when the Trigger statement is evaluated.



• **Property** – This statement type max exist independently of an **If/Else If/Else** item. A property statement is used to modify a specific property of an object using an expression of the form:

Object.Property = ExpressionText

Statements can be inserted by clicking their associated toolbar button or by dragging the toolbar button to the desired insert location.

• Clicking the button will attempt to insert the desired statement at the currently selected location within the condition tree. If the statement is unsupported (e.g. attempting to insert an **Else** statement without an **If** statement), then the insertion request will be ignored.

Conditions					×
Conditions 🖶 💥	Statements if elif elie	Operators = != < >	Commands 💽 Trigger 🔲 Property	/ 💥 🛛 Ŗ Evaluate	Clear Status
Condition 1	Condition 1				
	(Emp Add If St	atement			

• Similarly, dragging a statement will only allow you to drop at supported locations within the tree (e.g. an **Else** will only be droppable adjacent to an **If** statement that does not already have an **Else** attached).

Conditions						×
Conditions 🖶 💥	Statements if elif else	Operators = != < >	Commands 💽 Trigger 🔳	🛾 Property 🛛 🗱	Ŗ Evaluate	Clear Status
Condition 1	 Condition 1 (Empty) + 					

Conditions						×
Conditions 🖶 💢	Statements if elif else	Operators = != < >	Commands 💽 Trigger 🔲 Pr	operty 🞇	🚯 Evaluate	Clear Status
Condition 1	Condition 1 Condition 1 Condition 1 Condition 1 Condition 1 Condition 1					



If a statement requires an expression, then an editor will display once it is selected.

Conditions					×
Conditions 🖶 💥	Statements if elif elie	Operators = != < >	Commands 💽 Trigger 🔲 Property	💥 🛛 🖡 Evaluate Clea	ar Status
Condition 1	Condition 1				

The **If** and **Else If** statements require a valid Boolean expression. For example:

- Text1.PositionX < 100
- Text1.Opacity = Text2.Opacity
- Text1.Text == "ABC"

Conditions			×
Conditions 🕂 💥	Statements if elif else Operators = != < >	Commands 💽 Trigger, 🔲 Property 🙀 🛛 Ŗ Evaluate Clear Sta	atus
Condition 1	Condition 1	2	
	Text1.Text = "Hello"	Add Trigger Statement	
	(Empty)		

Trigger statements can be inserted independently (outside) **If/Else If/Else** statements or nested inside.

Statements if elif elee Operators = != < > Commands 🕥 Trigger 🔲 Property	🖌 🗱 Ev	aluate Clear Status
V 🔁 Condition 1		
If Text1.Text = "Hello"		
O		+
✓ IIII Actions		
Default		
Action 1		
✓ 2 Conditions		
Condition 1		
✓ I Scripting		
Condition1_ConditionEval		
🗸 🐑 External		
External Action Name		



Statements if elif else Operators = != < >	Commands 📀 Trigger 🔲 Prope	rty 🐹	🚯 Evaluate	Clear Status
✓ 2 Condition 1				
If Text1.Text = "Hello"				
Action: Action 1				+
✓ i⇒ Actions				
Default				
Action 1				
Conditions いろう				
Condition 1				
✓				
Condition1_ConditionEval				
✓				
External Action Name				

When evaluated, a **Trigger** statement will execute all of the checked triggers in the list. For example:

Conditions					×
Conditions 🖶 💥	Statements if elif else	Operators = != < >	Commands 🕘 Trigger 🔲 Propert	y 🗱 🛛 🕞 Evaluat	e Clear Status
Condition 1	Condition 1 Condition 1 First = "He Action 1 Action 2	ello"			

When executing the above condition, Prime will first evaluate whether the Text property of the Text 1 object is currently set to Hello. If the text matches, then Action 1 will be triggered. If the text does not match, then evaluation falls through to the Else statement, which in turn triggers Action 1.

This behavior can be verified by clicking the Evaluate button on the statement toolbar. Icons will appear indicating the evaluation path taken by the condition at the time of execution.

Conditions			×
Conditions 🖶 💥	Statements if dif die Operators = != < >	Commands 🕑 Trigger 🔲 Property	🗱 🛛 🖡 Evaluate 🛛 Clear Status
Condition 1	 Ze Condition 1 if Text1.Text = "Hello" Action 1 Action 1 Action 2 		

Clicking the Clear Status toolbar button will clear the evaluation icons from the window.



Conditions		×
Conditions 🕂 💥	Statements 🗊 💷 Operators = != < > Commands 🕑 Trigger 🗔 Property 💢	🚯 Evaluate 🛛 Clear Status
Condition 1	 Condition 1 ThisActionAlwaysExecutes If Text1.Text = "Hello" ThisActionExecutesIfTextIsHello If Text1.PositionX > 100 ThisActionOnlyExecutesIfBothTestsAreTrue ThisActionOnlyExecutesIfBothTestsAreTrue If Text1.Text == "ABC" ThisActionExecutesIfTextIsABC 	

Displayed above is a more complex condition tree.

Conditions						×
Conditions 🕂 💥	Statements if elif else	Operators = != < >	Commands 💽 Trigger	🗖 Property 🞇	🚯 Evaluate	Clear Status
Condition 1	Condition 1			<i>G</i> ³		
	💥 (Empty)			Add Property Sta	tement	

Property statements may be inserted independently (outside) **If/Else If/Else** statements or nestled inside.

Conditions		×
Conditions 🖶 💥	Statements 💷 💷 Operators = != < > Commands 🕑 Trigger 🔲 Property 💥	🕞 Evaluate Clear Status
Condition 1	✓ 22 Condition 1 ✓ if Text1.Text == "Hello" □ Text2.Text = "ABC"	

When evaluated, a valid **Property** statement will modify an object property with the result of the defined expression.

Valid **Property** statements have the form:

Object.Property = ExpressionText

For example:

Text1.Test = "ABC" + Text2.Text

In the above **Property** statement, the Text property of Text 1 will be set to the result of adding "ABC" to the Text property of Text 2.



Adding a Comment within a Condition, Logic, or Stylesheets

Typing an apostrophe ('), allows a comment to be added to Conditions, Logic, and Stylesheets which will not be executed. Comments can be used to give context to some logic and comment out logic statements.

티포ወ태포티					Active 🗕 🔤		+
Parameters / Expressio	ns						
Au Parameters	🕂 Add 🕶 💢 Remove 📄 Link Para	ameter 🕶 🚞 🕶 🔚 Save					
(0) Expressions	🛞 🖳 Name		Value			Bindings	
	nabc Parameter 1		Breaking News				
Logic							
2 Conditions	+ 🗙 🖿 🕞 🕴	Statements if elif else 📮	Commands 😔 Trigger	Property	Operators 🔀	" 📕 Evaluate	Clear Status
📦 Logic	Condition 1	'Use Single Quote to Add Cor	mments in Conditions				
Style Sheet		Parameter1.Value = "Breaking	g News"				
		Parameter I.Value = "This Lin	e will not Execute"				



In the example above:

- The first line doesn't have errors as it starts with a quote
- The second line is executed and changes parameter
- The third line is not executed and does not change parameters before it has a single ' in front.

Chyron

Triggering a Condition

Once the condition object has been configured appropriately, simply choose the condition when editing a trigger list anywhere within the editor. For example, it is possible to execute a condition whenever the Text Changed event of a Text object is raised.

			Playout
🕭 World 🛛 🗨 Auto Select	Properties Events	S	
	Text Changed Property Change Add Carlor Ren Property	Condition: Condition 1 Actions Condition 1 Action 1 Action 2 Conditions Condition 1 Scripting Text1_TextChanged External External External External Action Name	



Show what triggers a condition

To determine what triggers a condition, something we called "Triggered By" list click the lightning bolt on the Conditions dialog toolbar. This opens the "Triggers by" list.

This screen shot shows that the "Text1 TextChanged" event triggers Condition1 to execute.

Canvas Control Panel Scripting							
864	- 8	Condition	Triggers				×
21 6 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Stament I I I I Gamanda		Events Count 1 Events Count 1 Events Conditions Image: Conditions <td></td> <td></td> <td></td> <td></td>				
Condition 1	 If Text1.Text=="Hello" Action: Action 1 		Keyframes Count		-		
			Object	Name	Frame		
Timeline							
Default Action 1 🔥 Add Action							
Action 🦻 💘 📑 Trigger	ed By (0) P I I I II II I						
✓ Text 1	0:00 1:00 2:00 3:00					OK	Cancel



ABOUT US

Chyron is ushering in the next generation of storytelling in the digital age. Founded in 1966, the company pioneered broadcast titling and graphics systems. With a strong foundation built on over 50 years of innovation and efficiency, the name Chyron is synonymous with broadcast graphics. Chyron continues that legacy as a global leader focused on customer-centric broadcast solutions. Today, the company offers production professionals the industry's most comprehensive software portfolio for designing, sharing, and playing live graphics to air with ease. Chyron products are increasingly deployed to empower OTA & OTT workflows and deliver richer, more immersive experiences for audiences and sports fans in the arena, at home, or on the go.

CONTACT SALES

EMEA • North America • Latin America • Asia/Pacific +1.631.845.2000 • sales@chyron.com

