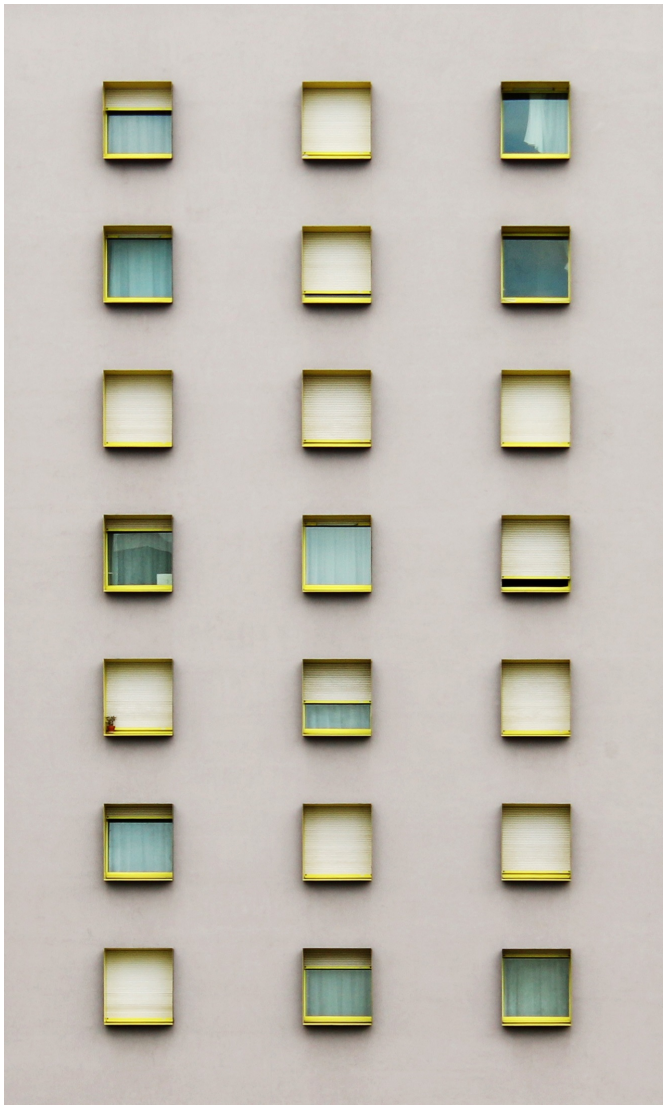
Continuity

Quick Reference Guides

Continuity Declarative Object Notation

Use this guide to get started with defining shapes using Continuity Declarative Object Notation.



Introduction

Continuity Declarative Object Notation “CDON”

Continuity Declarative Object Notation “CDON” is a simple C-like declarative language that can be used to define:

- **Custom Shapes** for individual Diagram Elements; and
- Shapes for **User Meta-Types** definitions that can used to create Elements in Continuity**.**

Custom Shapes

The appearance of any individual Diagram Element in Continuity can be altered and defined using a **Custom Shape script** which tells Continuity how to draw the Element’s representation on a given Diagram.

Any given Diagram Element’s Custom Shape script can be defined using the Diagram Element’s *Properties* window, on the *Shape -> Custom* Shape tab.

A Custom Shape can be defined using one of two techniques:

- Continuity Declarative Object Notation “CDON”, the subject of this Quick Reference Guide; or
- C# script, a functional language that uses Continuity’s internal API to perform drawing operations.

User Meta-Types

User Meta-Types are user-defined Element types that can be **configured directly** or **imported** into Continuity.

Configured User Meta-Types

The shapes for User Meta-Types that are configured directly into Continuity by an end-user are defined using the *Manage User Meta-Types* window. This is accessible from the *User Meta Types -> Manage Types…* context menu on the root *Repository* window.

A configured User Meta-Type’s shape can be defined using one of two techniques:

- Continuity Declarative Object Notation “CDON”, the subject of this Quick Reference Guide; or
- Free-hand drawn using the “Design Surface” tab on the *Manage User Meta-Types* window.

Imported User Meta-Types

User Meta-Types that are imported into Continuity are defined using XML import files. These files can be imported using the *User Meta Types -> Import from XML…* context menu on the root *Repository* window. These XML files contain the definitions of User Meta-Types including their CDON shape script.

Language Syntax

Continuity Declarative Object Notation "CDON" is a C-like declarative language. CDON syntax will be very familiar to anyone with experience using languages such as C, C++, C#, Java and ECMA Script, however it is simple and can be learned by people with limited or no technical skills.

Basic Constructs

The basic construct of a CDON script is the *object declaration*, which comprises an *object type* statement followed by a *block* that allows *object attributes* to be defined, for example:

```
Rectangle
{
}
```

Multiple object declarations can exist in a CDON script, in which case Continuity will draw each shape in turn over the top of each other in the order defined in the CDON script. For example:

```
Rectangle
{
}
Ellipse
{
}
Line
{
}
```

Attributes

Object declarations have *attributes*, which allow the available characteristics of the shape to be defined. An attribute comprise a *name* followed by a *value*, for example:

```
Rectangle
{
	Thickness : 2.0;
}
```

Attribute Sub-Objects

Some object types have attributes that are themselves *complex object types* rather than primitives such as decimal numbers or text. Such attributes comprise a *name* followed by a *sub-object declaration*, for example:

```
Line
{
	To : Coord
	{
		X : 0.0;
		Y : 1.0;
	}
}
```

Note also that some object types also allow multiple attributes with the same name, for example the *Geometry* attribute of an *Area* object declaration.

Line Endings

Note that attribute declaration lines with a primitive value (e.g. Decimal Number) end with " ; ".

Example A

The following simple example illustrates the syntax of CDON. This draws a ***rectangle*** shape with a border *thickness* of 2.0 and border *colour* of black, from the *location* coordinates 0%|0%, a *width* of 100%, a *height* of 100%, and a red *solid background* with 50% opacity.

```
#!DECLARATIVE/CDON
Rectangle
{
	Thickness : 2.0;
	Location : Coord
	{
		X : 0.0;
		Y : 0.0;
	}
	Width : 1.0;
	Height : 1.0;
	Foreground : Colour
	{
		Red : 0.0;
		Green : 0.0;
		Blue : 0.0;
	}
	Background : SolidBackground
	{
		Opacity : 0.5;
		Colour : Colour
		{
			Red : 1.0;
			Green : 0.0;
			Blue : 0.0;
		}
	}
}
```

Conditional Statements

The CDON language contains simple conditional constructs that allow you to change object declaration *attribute values* based on the values of attributes of the Element that is being drawn.

Conditional statements begin with a '#' symbol after which the condition is written enclosed within square brackets '[…]'. For example:

In the example above, the *Thickness* attribute of the shape declared is defined conditionally.

```
Rectangle
{
  Thickness : #[name='Test'?1.0|2.0];
  …
}
```

A conditional statement is made up of a *boolean test* (i.e. *name='Test'* in the example above) followed by a question-mark '?' symbol, then a *true value* (i.e. '1.0' above) and a *false value* (i.e. '2.0' above) separated by a pipe '|' symbol:

```
#[<boolean test>?<when true>|<when false>]
```

CDON supports two boolean test operators, being *equals* '=' and *not equals '!'*:

```
#[name='Test'?<when true>|<when false>]

#[name!'Test'?<when true>|<when false>]
```

Testable Element Attributes

The following Element attributes can be tested in a *conditional statement*:

name

description

notes

abstract

stereotype

For example:

```
#[name='T'?<when true>|<when false>]

#[description!'T'?<when true>|<when false>]

#[notes='T'?<when true>|<when false>]

#[abstract=true?<when true>|<when false>]
#[abstract=false?<when true>|<when false>]

#[stereotype='T'?<when true>|<when false>]
```

Object Declaration	Attribute	Multiplicity	Type	Comments
Area	Thickness	One	Decimal Number	Range between 0.0 and 10.0, representing the thickness of the border or line to draw
	Foreground	One	Colour sub-object	
	Background	One	Background sub-object (SolidBackground or GradientBackground)	
	Geometry (multiple)	One or Many	Coord sub-object	
Ellipse	Thickness	One	Decimal Number	Range between 0.0 and 10.0, representing the thickness of the border or line to draw
	Location	One	Coord sub-object	
	Width	One	Decimal Number	Range between 0.0 and 1.0, representing the % of the Diagram Element's on-screen **width** to draw
	Height	One	Decimal Number	Range between 0.0 and 1.0, representing the % of the Diagram Element's on-screen **height** to draw
	Foreground	One	Colour sub-object	
	Background	One	Background sub-object (SolidBackground or GradientBackground)	
Line	Thickness	One	Decimal Number	Range between 0.0 and 10.0, representing the thickness of the border or line to draw
	From	One	Coord sub-object	
	To	One	Coord sub-object	
	Foreground	One	Colour sub-object	
Rectangle	Thickness	One	Decimal Number	Range between 0.0 and 10.0, representing the thickness of the border or line to draw
	Location	One	Coord sub-object	
	Width	One	Decimal Number	Range between 0.0 and 1.0, representing the % of the Diagram Element's on-screen **width** to draw
	Height	One	Decimal Number	Range between 0.0 and 1.0, representing the % of the Diagram Element's on-screen **height** to draw
	Foreground	One	Colour sub-object	
	Background	One	Background sub-object (SolidBackground or GradientBackground)	

Object Declaration	Attributes	Multiplicity	Type	Comments	Example

Colour	Red	One	Decimal Number	Range between 0.0 and 1.0, representing the % of Red in the colour to be displayed	`Foreground : Colour`
					`{`
					`  Red : 0.0;`
					`  Green : 0.2;`
					`  Blue : 0.9;`
					`}`

	Green	One	Decimal Number	Range between 0.0 and 1.0, representing the % of Green in the colour to be displayed	
	Blue	One	Decimal Number	Range between 0.0 and 1.0, representing the % of Blue in the colour to be displayed	

Coord	X	One	Decimal Number	Range between 0.0 and 1.0, representing the X coordinate as a % of the Diagram Element's on-screen width.	`Location : Coord`
					`{`
					`  X : 0.5;`
					`  Y : 0.1;`
					`}`

	Y	One	Decimal Number	Range between 0.0 and 1.0, representing the Y coordinate as a % of the Diagram Element's on-screen height.	

Solid Background	Opacity	One	Decimal Number	Range between 0.0 and 1.0, representing the % opacity to use in displaying the background colour.	`Background : SolidBackground`
					`{`
					`  Opacity : 0.5;`
					`  Colour : Colour`
					`  {`
					`  Red : 0.1;`
					`  Green : 0.5;`
					`  Blue : 0.2;`
					`  }`
					`}`

	Colour	One	Colour sub-object		

Object Declaration	Attributes	Multiplicity	Type	Comments	Example
Gradient Background	OpacityFrom	One	Decimal Number	Range between 0.0 and 1.0, representing the % opacity to use in displaying the background colour from.	
	OpacityTo	One	Decimal Number	Range between 0.0 and 1.0, representing the % opacity to use in displaying the background colour to.	
	ColourFrom	One	Colour sub-object		
	ColourTo	One	Colour sub-object		
	Vector	One	Line sub-object	Line sub-object describing the direction and degree of the gradient between colour from and colour to.	

```
Background : GradientBackground
{
        OpacityFrom : 0.2;
        OpacityTo : 0.9;
        ColourFrom : Colour
        {
                Red : 0.9;
                Green : 0.6;
                Blue : 0.0;
        }
        ColourTo : Colour
        {
                Red : 0.9;
                Green : 0.6;
                Blue : 0.0;
        }
        Vector : Line
        {
                From : Coord
                {
                        X : 0.0;
                        Y : 0.0;
                }
                To : Coord
                {
                        X : 0.0;
                        Y : 1.0;
                }
                Thickness : 0.0;
                Foreground : Colour
                {
                        Red : 0.0;
                        Green : 0.0;
                        Blue : 0.0;
                }
        }
}
```

Example B

The following simple example illustrates the syntax required to draw a black diagonal **Line** from 0/0 to 1/1, with a thickness declaration based on the name of the Element being drawn being “A” or not.

```
#!DECLARATIVE/CDON
Line
{
	Thickness : #[name=‘A’?1|2];
	From : Coord
	{
		X : 0.0;
		Y : 0.0;
	}
	To : Coord
	{
		X : 1.0;
		Y : 1.0;
	}
	Foreground : Colour
	{
		Red : 0.0;
		Green : 0.0;
		Blue : 0.0;
	}
}
```

Example C

The following simple example illustrates the syntax required to draw an **Ellipse** at location 0/0 with a width of 100% and a height of 100%.

```
#!DECLARATIVE/CDON
Ellipse
{
	Thickness : 2.0;
	Location : Coord
	{
		X : 0.0;
		Y : 0.0;
	}
	Width : 1.0;
	Height : 1.0;
	Foreground : Colour
	{
		Red : 0.0;
		Green : 0.0;
		Blue : 0.0;
	}
	Background : SolidBackground
	{
		Opacity : 0.5;
		Colour : Colour
		{
			Red : 1.0;
			Green : 0.0;
			Blue : 0.0;
		}
	}
}
```

Example D

The following simple example illustrates the syntax required to draw an **Area** with geometry running from 0/0 to 0/1 to 1/1 to 1/0 and back to 0/0.

```
#!DECLARATIVE/CDON
Area
{
	Thickness : 2.0;
	Geometry : Coord { X : 0; Y : 0; }
	Geometry : Coord { X : 0; Y : 1; }
	Geometry : Coord { X : 1; Y : 1; }
	Geometry : Coord { X : 1; Y : 0; }
	Foreground : Colour
	{
		Red : 0.0;
		Green : 0.0;
		Blue : 0.0;
	}
	Background : SolidBackground
	{
		Opacity : 0.5;
		Colour : Colour
		{
			Red : 1.0;
			Green : 0.0;
			Blue : 0.0;
		}
	}
}
```